REPORT DOCUMENTATION PAGE	Form Approved OMB NO. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE 19 March 2003	3. REPORT TYPE AND DATES COVERED Final Report 01 Aug 02 ~~September 1, 2002~~ through January 31, 2003

4. TITLE AND SUBTITLE *GASP - Generator for Adaptive Statistical Pattern Recognition Systems* STTR 2002.1 Topic Army02-T004 Topic Title: Analysis and Characterization of Pattern Classifiers	5. FUNDING NUMBERS DAAD19-02-C-0067
6. AUTHOR(S) Mr. Gary Key, PI, Frontier Technology, Inc. Dr. Gerhard Ritter, University of Florida Dr. Mark Schmalz, University of Florida	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Frontier Technology, Inc. 5205 Leesburg Pike, #1110 Falls Church, VA 22041 Attention: Gary Key	8. PERFORMING ORGANIZATION REPORT NUMBER Proposal 44157-C1-ST1
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) U. S. Army Research Office P.O. Box 12211 Research Triangle Park, NC 27709-2211 Attention: Dr. William Sander	10. SPONSORING / MONITORING AGENCY REPORT NUMBER 44157.1-C1-ST1

11. SUPPLEMENTARY NOTES

The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by other documentation.

12 a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution unlimited.	12 b. DISTRIBUTION CODE

20030401 016

13. **ABSTRACT (Maximum 200 words)**

Report developed under STTR contract for topic ARMY02-T004

Under this effort, Frontier Technology, Inc. (FTI) and University of Florida (UF) are developing designs for automatically-generated statistical pattern recognition systems (GASPs) that can classify uncooperative targets among time-varying natural and manmade backgrounds. We also propose to analyze the performance of the envisioned GASPs to:

(a) covertly acquire feature data (e.g., statistical, spectral, and spatial cues) from target/background imagery,
(b) apply multiple classifiers to target/background information to select probable target location and identity,
(c) apply inferencing rules to disambiguate infeasible or contradictory classifier outputs.

Pattern selection, key to successful system operation in mission- and threat-specific scenarios, will utilize Dempster-Schaefer theory and UF's powerful data fusion paradigm, Morphological Neural Nets (MNN).

Phase-I will evaluate, extend and exploit FTI and UF's successful, DoD-sponsored R&D for dynamic pattern recognition and ATR to develop and test an efficient system design for target cla ssifier output fusion and disambiguation. System design will include analysis of complexity and cost of potential hardware implementations. In a Phase-II effort, we will use Phase-I results to drive candidate pattern downselection in FTI's DoD-supported TNE paradigm. MNNs and TNE have been proven highly successful in a wide variety of recognition problems, thus we propose to analyze GASP system performance in realistic ATR scenarios.

14. SUBJECT TERMS STTR REPORT morphological neural nets, TNE Paradigm automatic target classifiers data fusion			15. NUMBER OF PAGES 72
			16. PRICE CODE
17. SECURITY CLASSIFICATION OR REPORT **UNCLASSIFIED**	18. SECURITY CLASSIFICATION ON THIS PAGE **UNCLASSIFIED**	19. SECURITY CLASSIFICATION OF ABSTRACT **UNCLASSIFIED**	20. LIMITATION OF ABSTRACT **UL**

NSN 7540-01-280-5500 **Standard Form 298**
(Rev.2-89)
Prescribed by ANSI Std. 239-18

GASP – Generator of Adaptive Statistical Pattern Recognition Systems

Contract #DAAD-19-02-C-0067
Phase-I STTR Army02-T004

FINAL REPORT

31 January 2003

Gary Key, PI
gkey@fti-net.com, ph321-277-8396

Mark S. Schmalz, UF Lead
mssz@cise.ufl.edu, *ph*: 352-392-6831

Gerhard X, Ritter, Co-UF Lead
ritter@cise.ufl.edu, *ph*: 352-392-1212

Frontier Technology, Inc.
6785 Hollister Avenue
Goleta, CA 93117

University of Florida
Center for Computer Vision and Visualization
University of Florida, Gainesville, FL 32611-6120

Submitted to:

U. S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211
Attention: Dr. William Sander

Abstract

In this Phase I STTR effort, Frontier Technology Inc. (FTI), and University of Florida (UF) are teamed to develop and analyze designs for automatically ***generated adaptive statistical pattern recognition*** (**GASP**) systems to detect targets against time-varying natural and manmade backgrounds. This research has been performed in the context of FTI's DoD-supported **"TNE"** *("Tabular Nearest-neighbor Encoder")* paradigm for classify-before-detect (CBD) pattern recognition and ATR. UF's powerful data fusion paradigm, Morphological Neural Nets **(MNNs)** has also been analyzed for this application. MNN's and TNE have been proven highly successful in a wide variety of recognition problems. Following preliminary design, FTI and UF analyzed GASP system performance to demonstrate increased probability of detection (Pd) with respect to MNNs or more traditional Bayesian systems alone. Design for real-time or near-real time implementations of GASP included analysis of state-of-the-art embedded processing technologies such as field programmable gate arrays (FPGAs) and single-instruction multiple-data (SIMD) array processors.

FTI's TNE paradigm is built around a very low complexity, vector search/comparison algorithm based completely on Boolean operations among binary-component, data-related vectors stored in a large table. FTI has successfully applied the process to several previous applications with considerable success. Here, it forms the architectural basis for our approach to the derivation of time-evolving inferences through the fusion of multiple sensor/ classifier responses and the comparison of these to a vectorized knowledge base or training set. Application of the TNE process on any (and all) signature vectors results in clusters of non-isolated "**agreement events**" with regard to the components of every vector in the training set, at the cost of only a relatively small number of binary "***and***" and "***or***" processor operations within the large table. We generate a binary (Boolean) element matrix "**Agreement Map (AM)**" for each test vector (e.g., image patch), and then consider all these related AMs in the inference process. We can even significantly suppress random noise "agreements" by selecting only those agreement events that occur along with another contiguous (e.g. adjacent pixels) and/or contemporaneous (e.g., a single pixel in subsequent samples or frames) event of the same type. This is also readily doable via a relatively small number of Boolean operations. Finally, we consider associations among training set exemplars. As a result of variation in object orientation, size, articulation, configuration, etc.; a target is best represented in the training set by a (possibly large) collection of exemplars. We can gain a measure of "inference strength" from a cluster of **agreement events** within one of these target-related exemplar clusters by applying the Boolean "***or***" operation on the respective AMs. Thus, our TNE paradigm facilitates the generation of target-related inferences based on the fusion of (perhaps) many diverse observations as compared to collections of candidate exemplars. The final results of the process are clusters of **agreement events** indexed by training set exemplar group number, vector component (e.g., pixel), and collection time (or spectrum, or however the data vectors themselves are indexed). As a part of this Phase I contract, we have implemented and partially optimized the above process.

We also considered statistical clustering and MNN approaches to further interpret the composite "training set agreement event space" (i.e., a "stack" or related AMs) in order to generate inferences with respect to the viewing objects. A comment: we believe that our approach, a strongly "classify-before-detect" one, is architecturally similar to the human visual recognition system.

Table of Contents

1.0 Introduction

Frontier Technology, Inc. and University of Florida (UF) are pleased to submit this Final Report pursuant to Army STTR 02-T004, Analysis and Characterization of Pattern Classifiers.

Military and domestic targets in digital imagery can be detected by a variety of techniques, including hyperspectral imaging, change detection, and a combination of broadband signature and spatial pattern recognition algorithms. Hyperspectral imaging acquires dense signature information across numerous spectral bands, which can be processed with techniques such as component selection to maximize target-background separation. By locating areas where target camouflage statistics (e.g., hue, patterning, and changes or motion) do not match background statistics, candidate targets can be identified. It is well known that further decomposition of spatial and spectral attributes of candidate targets using adaptive filtering or classification techniques can yield estimates of target presence at a given location.

Image-based automated target recognition systems are typically limited by natural effects such as statistical, spectral, and geometric similarity of targets or target-specific features viewed against a variety of naturally-occurring backgrounds. In dynamic ATR imagery, the movement of targets among cover objects (e.g., overhanging foliage, rocks, etc.) can obscure targets sufficiently to confound tracking algorithms. A special type of image processing operation or ATR filter, called an anomaly detector, is able to highlight regions of an image or video sequence that's statistical, spectral, or geometric properties (including spatial frequency) do not match those of its proximal background. ATR systems, which usually comprise a suite of ATR filters and one or more anomaly detectors, can be used to detect candidate target regions by mathematically combining (fusing) the filter outputs, which can then be processed by one or more pattern classifiers to yield an estimation or prediction of target location or identity. This ATR strategy is called detect-before-classify (DBC).

Unfortunately, DBC ATR systems have several significant deficits. Firstly, accurate target classification is predicated on the target having been successfully detected and accurately represented by a candidate target blob. Secondly, the ATR filters are assumed to be selected via *a priori* knowledge of salient target characteristics. Thirdly, one assumes that the classifiers will generate outputs that are not mutually contradictory in the context of ground truth. For example, if classifier C_1 claims a tank is detected, but C_2 or C_3 respectively report that a jeep or bus is detected, then the classifier output will be a less reliable estimate of actual target identity. Fourthly, if the ATR filter input (e.g., multispectral imagery) has statistical, spectral, or other attributes that vary spatiotemporally in the secular sense (e.g., beyond mere intra- or inter-frame variations), then the classifier output reliability will drift as a function of time. This typically renders the ATR system output nonstationary if the system design is based on nonadaptive ATR filter and classifier algorithms.

Another limitation of detect-before-classify ATR systems is sensitivity of the detection stage to noise, spatial distortion, and spectral shifts in the sensor response function. These are but a few deficits of realistic sensing systems that can derive (for example) from poor hardware design; detector or amplifier malfunction in the presence of environmental hazards; sensor platform roll, pitch, or yaw; and optical component response perturbations given mechanical, thermal, or hydration stress. Numerous other problems are possible, and have been discussed elsewhere [Sch96]. In summary, if the detection stage is sensitive to noise, other systematic error, or mission-specific perturbations, then the information contained in the candidate target blobs is likely to be a less accurate descriptor of target content. As a result, probability of detection (Pd) can decrease and rate of false alarms (Rfa) can increase, at significant liability to missions depending on such systems.

In this Phase-I STTR, we address two solutions to the aforementioned problems. Firstly, we have developed a an approach to making classifier output more reliable when (a) individual classifier design is based on concepts of Bayesian statistics (where significant a priori information is required); and (b) the characteristic response of multiple classifiers employed in an ATR system is known. This has traditionally been achieved by applying evidential reasoning to multiple classifier outputs, as illustrated notionally in Figure 1.1. We studied three competitive technologies for classifier output disambiguation: Dempster-Schaefer theory, fuzzy logic, and an innovative paradigm called Morphological Neural Nets (MNNs). The best-performance technology would be selected for system design and prototype implementation.

Secondly, per the Phase-II goal of generating automated statistical pattern recognition systems (GASP) we note that there exists a much more effective technique for ATR, called *classify-before-detect* (CBD) which efficiently combines detection and classification in a concurrent operator (similar to the human vision system). CBD systems tend to be more noise tolerant, and can adapt to nonstationary input using *a priori* or *a posteriori* knowledge including a history of filter performance under various input conditions. It is important to note that FTI and UF have developed both DBC and CBD systems for a wide variety of ATR applications for numerous DoD-funded projects, and thus have significant research and development expertise in this area. It is also important to observe that, throughout this report, we refer to Phase-II developments and results in order to tie our Phase-I results into the broader GASP system concept.

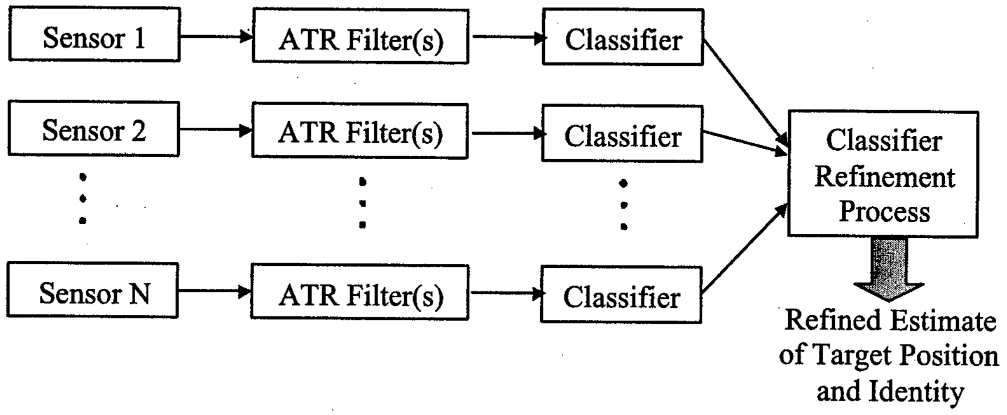


Figure 1.1: *Schematic diagram of multi-sensor processing and classification architecture, in support of target detection*

In contrast to model-based, noise-limited approaches developed by previous DBC research efforts (whether or not statistical pattern recognition was employed), we have developed a broad-based system design for GASP that would initially configure an exemplar pattern database using a morphological NN. The resultant *codebook* of statistical target templates would be based on significant prior knowledge of target attributes, both before and after the ATR filtering process. Given this pattern database, FTI's TNE pattern recognition algorithm can be used to classify input data (e.g., multispectral imagery) *immediately following the sensor output*, thus implementing classify-before detect.

Our research team is highly qualified to address all aspects of this research and development project. The Principal Investigator, Mr. Gary Key, has over 30 years experience in data/signal exploitation and sensor/processor characterization. He has led numerous research efforts in these areas. Mr. is the inventor/discoverer of the TNE Paradigm described in this Final Report. The Principal UF Lead, Dr. Mark Schmalz, directed system design and evaluation at UF, and was responsible for all aspects of the research effort. Dr. Schmalz has over 15 years experience in the design and implementation of large-scale software systems for physical and optical modeling, as well as target recognition and image enhancement. Dr. Schmalz' research in target recognition and mine detection has recently been funded by the Air Force (AFOSR) and Navy (ONR), and emphasizes detection of natural or manmade targets (e.g., landmines or vehicles) obscured by overlying cover, refractive or scattering media (e.g., submerged targets or sea mines), or atmospheric obscurants. The UF Co-Lead, Dr. Gerhard X Ritter, assisted in theory development and algorithm design. Dr. Ritter is Professor of Computer and Information Science and Engineering, Professor of Mathematics, and Director of the Center for Computer Vision and Visualization at University of Florida. Dr. Ritter has over 15 years experience in DoD-funded development of theory, algorithms, and software for computer vision applications, including human/vehicle detection, tracking, classification, and recognition.

The GASP project has yielded significant technological advances that can be applied to areas such as military and domestic object detection applications, for example, battlefield detection of concealed defense assets, and (using technology developed in this and other research efforts) monitoring of battle areas for groups or structured arrays of concealed vehicles or hazards. Adaptations of the proposed algorithms and software could be employed in domestic applications such as law enforcement (e.g., contraband detection), as well as environmental or wildlife management studies where passive surveillance or monitoring of animal or plant subjects is required.

1.3. Project Scope and History

This research project involved efforts in: (a) analysis of multi-classifier pattern recognition systems for ATR, (b) adapting TNE and MNN algorithms for detecting targets embedded in naturally-occurring backgrounds; (c) comparative analysis of Bayesian rule-based classifiers, MNNs, and TNE alone for combining the results of multiple classifiers to support ATR with increased Pd and reduced Rfa; (d) performance analysis of classifiers in c), above, that could be implemented in Phase-I; (e) design or evaluation of advanced classifier processing architecture concepts using MNNs to process the TNE "Agreement Map"; and (f) development of demonstration capabilities and a Phase-II proposal in support of follow-on research.

1.4. Schedule of Work (SOW), with Conformance Data

The Schedule of Work (SOW) for the effort is listed below, with conformance or exceptions as indicated.

Task 1. Survey Evidential Reasoning techniques and data for GASP

Conformance: This task was completed as proposed, on schedule, as described in the following subtasks.

ST1.1. Specify requirements for target classifier input/output.

Conformance: Requirements for each post-sensor classifier input were specified as a vector of Boolean or grayscale values that represented sensor output, for example, spectral coefficients pertaining to target or background regions. Requirements for the post-sensor classifier output emphasized a classification score within the real-valued interval [0,1], or a Boolean score (e.g., target or no-target). These scores would then be input to the classifier refinement algorithm, which would produce a list of probable target identities, or an estimate of the probability of target presence given a hypothetical target identity.

ST1.2. Survey literature of evidential reasoning for newest developments.

Conformance: The available literature at the time of contract award was surveyed, and we did not find any significant new research developments since the date the Phase-1 proposal was submitted.

ST1.3 Select best-performance technologies on the basis of accuracy, adaptability, and efficiency

Conformance: We surveyed textbook algorithms for Bayesian inference and found these to be representative of rule-based classification algorithms. Morphological Neural Networks are superior to other neural network algorithms for the type of classification envisioned for a GASP system. FTI's TNE algorithm has been demonstrated to exhibit superior performance for template matching tasks, versus other types of template matching algorithms studied in previous research.

Task 2. Design and analysis of algorithms, software, and test imagery to support Classifier Output Fusion and Refinement for GASP

Conformance: This task was completed without Subtask 2.3 being performed for TNE (although error analysis was provided for Bayesian and MNN based classifiers).

ST2.1. Extend classifier performance analysis to include metrics for moving and stationary targets in video and still imagery.

Conformance: This subtask was completed by extending the Fd and Ffa pixel fraction metrics that UF developed in previous research to portray multiframe targets. These metrics are discussed in the Advanced Technology Summary (Section 7).

ST2.2. Specify enhancements to existing FTI and UF TNE and MNN algorithm descriptions and software for pattern classifier refinement discussed in this proposal. Incorporate UF's Morphological Neural Networks into the TNE Paradigm to (a) select the TNE codebook per mission-specific requirements and a history of classifier performance, and (b) select pattern clusters for TNE agreement map processing to maximize Pd and minimize Rfa.

Conformance: This UF subtask was redirected at the request of Frontier Technology's POC Mr. Key to emphasize MNN-based processing of the TNE agreement map, with research results summarized in Section 6.

ST2.3. Prototype the design of the classifier refinement system, and conduct noise, sensitivity, and complexity analysis to support use of GASP in the presence of sensor noise, systematic error, and nonergodic input.

Conformance: This subtask was completed for Bayesian classifiers (Section 4) and MNNs (Section 5). The MNN-specific theory allows one to construct kernel vectors that are robust in the presence of erosive or dilative noise. Since MNNs are robust classifiers in the presence of other types of noise, this extends the utility and accuracy of MNNs to provide not only classification, but classifier refinement in the presence of noise or systematic effects such as thresholding error in the TNE agreement map.

Task 3. End-to-end system design and Classifier Refinement Testing [E.3.2]

Conformance: This UF task was partially completed, but within the constraints of the Phase-1 proposed effort, which did not include consideration of the effect of MNNs on the TNE template database. With the agreement of FTI's POC Mr. Key, we also allocated a portion of the effort for this task to the theoretical development of MNN-enhanced TNE agreement map processing.

ST3.1. Design GASP system, design test procedures, and simulate performance using models and algorithms developed in Tasks 1 and 2.

Conformance: Performance and complexity of MNN-enhanced TNE agreement map processing were analyzed theoretically, which was simplified by the fact that MNNs converge in one pass of the network, rather than oscillating or reverberating as do classical NNs. Additionally, we re-examined the complexity of TNE.

ST3.2. Analyze sensitivity of system to dynamic targets and target-background motion Include effects of nonergodic input on classifier refinement algorithm learning rate, as well as performance measures such as Pd and Rfa.

Conformance: This subtask was completed for TNE, primarily at FTI, with technical assistance provided by UF. UF is continuing this work in the context of another DoD-funded project on which UF has a subcontract with FTI as prime contractor. Further discussion of these issues can be found in the Advanced Technology Summary (Appendix B).

Task 4. Management, Reporting, and Marketing Plan

Manage overall effort, developing technical reports, final report and internal and external review meetings.

Conformance: FTI (with the assistance of UF) has submitted monthly technical reports, and has written this final report. We are also collaborating with UF in writing the Phase-II proposal, at the request of the sponsor.

2.0 GASP System Concept

In this section, we review the concept development that was designed to support our Phase-1 research for Army STTR 02-T004, Analysis and Characterization of Pattern Classifiers. Section 2.1 contains a discussion of project objectives and system functionality, Section 2.2 overviews the technique of decomposition that supported GASP related research in multi-paradigm pattern recognition approaches, and Section 2.3 summarizes implementational issues, with concentration on challenges that are germane to the proposed Phase-II research and development effort.

2.1. Project Objectives and Achievements

Consider a multi-classifier system as shown, for example, in Figure 1.1. Here, a sensor suite comprised of N sensors (e.g., radar, electro-optic, inertial or GPS) forms a projection of the three-dimensional scene within the footprint of the sensors. This projection can contain intensity, spatial, range or spectral information, whose content and format is idiosyncratic to each sensor or sensor type, as well as to each target or background object. In order to process each sensor's output in a way that is accurate in and relevant to application-specific objectives, each of the N sensor outputs is typically input to a postprocessing computation that is often followed by a classifier. Alternatively, several postprocessed outputs can be combined prior to classification. The result is that $M \leq N$ classifier outputs are available, each of which can represent different estimates of target presence, for example, target identity, probability of detection or false alarm rate. In an ideal classification scenario, classifier output would also be associated with different data quality measures and degrees of confidence or caution.

The combination of these disparate classifier outputs is fraught with difficulty, primarily due to different parameters (e.g., a spectral detection process versus a time-of-flight based detection) and data quality (e.g., high computational error corrupting one classifier output, while another classifier is sensitive to image detector noise propagated through one of the aforementioned postprocessing computations). Various techniques such as linear or nonlinear combination, Bayesian or unsupervised classification, and supervised learning techniques such as neural networks have been proposed for refining the output of individual classifiers or combining such outputs to produce a more accurate classification.

The Topic Description for this STTR effort specified classifier refinement, particularly comparison with Bayesian approaches, as the preferred emphasis. Accordingly, we have considered Bayesian rule-based networks (Section 3), Frontier Technology's TNE pattern recognition paradigm (Section 4), and UF's Morphological Neural Networks (MNNs) pattern recognition paradigm (Section 5) for both classifier and post-classifier processing. Furthermore, we have investigated and formulated approaches for processing the TNE agreement map using MNNs (Section 6). The latter techniques represent important new developments in pattern recognition, which promise greater accuracy and information storage capacity, as well as improved refinement of classified sensor output.

2.2. System Decomposition

Consider again the multi-classifier system, in Figure 1.1. Here N sensors produce N output datastreams that are processed by ATR filters to extract salient features. The features are then classified, to yield preliminary estimates of target location or identity. In many realistic cases, the individual classifiers can produce conflicting results, due to input noise, sensor error or noise, partial information, or lack of coverage of the training set. Thus, a classifier refinement process is applied to the individual classifier outputs to produce a refined estimate of target location and an estimate of target identity, insofar as theory, algorithms, and training data permit. The salient parts of this system are thus

- *Sensor and Sampling Process(es)*, which can include source of noise and error, for example, due to image compression and decompression, undersampling, or quantization;
- *Feature Extraction Process(es)* that might or might not extract features accurately, or in a manner that provides optimally useful information to the follow-on classifier(s);
- *Feature Classification*, which can be erroneous due to incomplete coverage, lack of contextual information, sensitivity to error in the feature extraction process, etc.;
- *Classifier Output Refinement*, which is expected to compensate for the aforementioned noise and error effects, to provide a more accurate estimate of target location and identity.

In Phase-I research; we neglected the sensor and sampling effects, for two reasons. First, it was expected that the pattern recognition paradigm(s) developed under the scope of this study would be independent of sensor, sampling, and feature extraction effects. Second, time and resources did not permit end-to-end sensor modeling, which was not specified in the SOW. As a result, we concentrated on pattern recognition only, in the context of classifier refinement.

2.3. Implementational Issues

When considering the system diagram shown in Figure 1.1 in relationship to the three pattern recognition paradigms studied in this project, it is reasonable to determine how each paradigm can be decomposed in relationship to Figure 1.1 and the system decomposition discussed in Section 2.2. We have found that Bayesian rule-based classification covers the *classifier* and *classifier output refinement* blocks of Figure 1.1, while FTI's TNE paradigm covers all post-sensor processing, including sampling of the sensor data. In particular, TNE has as its prior information (in the Bayesian sense) the target template database, distance measures used to compute the matching score between a given sensor output sample and a target template, and the thresholding function and parameters employed in transforming the match score into a binary number that becomes a value in TNE's Agreement Map.

Morphological Neural Networks simplify the processing cascade shown in Figure 1.1 by directly operating on the sensor output, although MNNs can also be applied to feature detector output. Classifier refinement is not required when MNNs solely are employed. When MNNs are combined with TNE, then the MNN is used to process the Boolean data in the TNE Agreement Map. If the columns of the AM are viewed as preliminary classifier results for a given sample, then an MNN applied to the AM can produce a refined, target-specific estimate of presence; location data is provided by the TNE sample coordinates.

Computational complexity is an issue for all three paradigms studied. Bayesian rule-based pattern recognition requires a minimum of **O**(log N) time for runtime. Build-time (rule assembly) work varies with the complexity of the feature vector and training set size. TNE requires considerable build-time overhead to construct the binary pointer table from which the agreement map is derived. However, the run-time classification is I/O-intensive primarily, as discussed in Section 4. For example, given an N-point sample and a template database comprised of M targets, **O**(MN/W) I/O operations are required to build each AM on computers with maximum W-bit integer arithmetic (usually W=32 or W=64), while each target vector can be post-processed in the classifier refinement step in **O**(N/W) time. Since TNE is a massively parallel pattern recognition paradigm, these complexities can be straightforwardly reduced through the use of multiple parallel buses and processing elements, as discussed in [Key99]. MNNs are the most efficient of the three paradigms studied herein: build-time activity pertains directly to computation of the MNN weight matrix, which requires very small training times. Runtime classification is accomplished in one pass of the net: for an N-input net, this amounts to **O**(N) computation on an N-processor machine.

As shown in previous research [Key99, Rit98], both TNE and MNNs are tolerant of input noise and missing information. This claim cannot be made globally for Bayesian classifiers, where one must know all *a priori* probabilities, and be able to completely and accurately calculate the a posteriori probabilities for the entire expected target set. Additionally, MNNs can perform a limited amount of interpolation to support classification with partial information, while neither TNE nor Bayesian rule-based classifiers are capable of interpolation.

As a result of our Phase-I research; we envision the following implementational challenges in a Phase-II research effort:

- Quantifying the performance of TNE, MNNs, and the TNE/MNN combination in the presence of input noise, error, and partial information;
- Quantifying the performance limits on MNN-directed processing of the TNE agreement map, to determine limits of utility on TNE/MNN versus MNN or TNE alone;

- Effectiveness with which MNNs can be used to cluster target exemplars in TNE's template database, to achieve more efficient pattern recognition without degrading Pd or Rfa performance; and
- Hardware design and implementation for TNE/MNN applications, to achieve optimal or near-optimal space/time/error tradeoffs for various embedded processing scenarios.

We next provide a theoretical overview of Bayesian classifiers, with emphasis on classifier error and performance metrics.

2.4. References

[Key99] Key, G., M.S. Schmalz, F.M. Caimi, and G.X. Ritter. "Performance analysis of tabular nearest neighbor encoding algorithm for joint compression and ATR", in *Proceedings SPIE* 3814:115-126 (1999).

[Rit98] Ritter, G.X., P. Sussner, and J.L. Diaz de Leon. "Morphological associative memories", *IEEE Transactions on Neural Networks* 9(2):281-293 (1998).

3. Bayesian Rule-Based Pattern Recognition Paradigm

Bayesian decision theory is a statistical pattern classification approach that is based on two strong assumptions: (1) a given decision or classification problem can be posed in probabilistic terms, and (2) all of the relevant probabilities are known. Without initially considering either epistemological basis or realism of these assumptions, we first present a mathematical description of Bayesian classifier theory (Section 3.1). We then discuss techniques and metrics by which the performance (e.g., classification accuracy) of pattern classifiers in general, and Bayesian classifiers in particular, can be quantified (Section 3.2). Section 3.3 contains an overview of Bayesian classifier refinement techniques. Finally, in Section 3.4, we note the shortcomings of the Bayesian paradigm relative to statistical techniques such as FTI's TNE paradigm or neural networks.

3.1. Mathematical Description of Bayesian Pattern Recognition

Let us begin by considering a simple problem of classifying candidate targets in digital monochromatic imagery. In particular, assume that a given target area in an image (also called an *AOI*) can be classified according to its brightness or grayscale value *g*, as *target* ($g = g_1$) or *non-target* ($g = g_2$). We generalize this assumption by saying that there is an *a priori probability* $\Pr(g_1)$ that the current AOI is a target and some a priori probability $\Pr(g_2) = 1 - \Pr(g_1)$ that it is not a target. If the decision process is automated, then the simplest approach is to decide that a given AOI is a target if $\Pr(g_1) > \Pr(g_2)$; and conversely a non-target object. In general, the probability of error is the smaller of $\Pr(g_1)$ and $\Pr(g_2)$.

In practice, we view grayscale intensity *g* as a continuous random variable whose distribution depends upon multiple constraints (e.g., target size, shape, and surface texture or color; lighting angle(s) and color(s); environmental variables such as atmospheric temperature, humidity, or windspeed in the case of infrared imagery; camera lens, detector, and amplifier spatial and intensity response, etc.) Let $p(g \mid g_i)$ denote the *state-conditional probability density function* for *g*, given that the state of the target recognition environment is g_i, where I = 1 or I = 2 is possible in our highly simplified example. Then, the difference in brightness between target and non-target objects is described by

$$\Delta g = p(g \mid g_1) - p(g \mid g_2) .$$

Now let us make this problem more realistic by assuming that we know $\Pr(g_1)$ and $\Pr(g_2)$ as well as $p(g \mid g_1)$ and $p(g \mid g_2)$. Additionally, suppose we can measure g for a given AOI. *Bayes' Rule* shows that observing the value of g changes the a priori probability $\Pr(g_i)$ to the a posteriori probability $\Pr(g_i \mid g)$, as follows:

$$\Pr(g_i \mid g) = \frac{p(g \mid g_i) \cdot \Pr(g_i)}{p(g)}, \tag{3.1}$$

where the composite probability (also called mixture density) $p(x)$ is given by

$$p(x) = \sum_{i=1}^{2} p(g \mid g_i) \cdot \Pr(g_i). \tag{3.2}$$

If we have a value g for which $\Pr(g_1 \mid g) > \Pr(g_2 \mid g)$, then it is reasonable to decide that a target is present; conversely, a non-target object is recognized.

Now let us consider the probability of error in this simple example. Whenever we measure a particular grayscale value g, the probability of an error ε is given by

$$\Pr(\varepsilon \mid g) = \begin{cases} \Pr(g_1 \mid g) & \text{if AOI is classified as } \textit{target} \\ \Pr(g_2 \mid g) & \text{if AOI is classified as } \textit{non-target} \end{cases}. \tag{3.3}$$

As before, we can minimize the probability of error by deciding *target* when $\Pr(g_1 \mid g) > \Pr(g_2 \mid g)$, and *non-target* conversely. This estimation is correct because the average probability of error is given by

$$\Pr(e) = \int_{-\infty}^{\infty} \Pr(e, x)\,dx = \int_{-\infty}^{\infty} \Pr(e \mid x) p(x)\,dx. \tag{3.4}$$

If, for every value of g, $\Pr(e \mid g)$ is minimized, then the preceding integral is minimized.

There are well-known arguments that the scale factor $p(x)$ is inherently unimportant because it merely ensures that the a posteriori probabilities sum to unity. Elimination of $p(x)$ yields the following Bayesian decision rule:

Classify AOI as *target* if $p(g \mid g_1)\Pr(g_1) > p(g \mid g_2)\Pr(g_2)$, otherwise classify as *non-target*.
Let us now consider these concepts in light of the two-class Bayesian decision problem inherent in *target* versus *non-target* discrimination.

3.1.1. Two-Class Problem. Let $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ denote a pattern vector comprised of n features, which is to be classified in class c_1 or c_2. Denoting the a posteriori probability of c_1 given $\mathbf{x}$, we have previously observed that, in the simplest classifier based upon probabilities only, $\mathbf{x}$ is in class c_1 if $\Pr(c_1 \mid \mathbf{x}) > \Pr(c_2 \mid \mathbf{x})$ or, conversely, in c_2. In Equation (3.1), we showed that the a posteriori probability $\Pr(c_i \mid \mathbf{x})$ can be calculated from the a priori probability $\Pr(c_i)$, for i = 1 or i = 2, using the Bayes rule:

$$\Pr(c_i \mid \mathbf{x}) = \Pr(c_i)\, p(c_i \mid \mathbf{x}) \,/\, p(\mathbf{x}),$$

where $p(\mathbf{x})$ denotes the mixture density function referred to as a scale factor in the preceding discussion. Since $p(\mathbf{x})$ is positive and common to both sides of the inequality inherent in Bayesian decision, we can write the decision rule as

$$\Pr(c_1)\, p(c_1 \mid \mathbf{x}) \underset{c_2}{\overset{c_1}{\gtrless}} \Pr(c_2)\, p(c_2 \mid \mathbf{x}). \tag{3.5}$$

This can also be expressed in terms of a *likelihood ratio* $L(\mathbf{x})$, defined as

$$L(\mathbf{x}) = \frac{p(c_1 \mid \mathbf{x})}{p(c_2 \mid \mathbf{x})} \underset{c_2}{\overset{c_1}{\gtrless}} \frac{\Pr(c_2)}{\Pr(c_1)}, \tag{3.6}$$

where $\Pr(c_2)/\Pr(c_1)$ is called the *threshold value of the likelihood ratio* for the decision. Writing this in logarithmic form, we have the *minus log likelihood ratio*

$$h(\mathbf{x}) = -\log(L(\mathbf{x})) = -\log(p(c_1 \mid \mathbf{x})) + \log(p(c_{21} \mid \mathbf{x})) \underset{c_2}{\overset{c_1}{\gtrless}} \log\left(\frac{\Pr(c_1)}{\Pr(c_2)}\right) = \log(\Pr(c_1)) - \log(\Pr(c_2))$$

which is also called the *discriminant function*. The comments pertaining to decision error, which surround Equations (3.3) and (3.4), also apply to this discussion.

3.1.2. Implementational Issues. It is possible to compute h given specific assumptions about prior and posterior probabilities. For example, when the values $p(c_i \mid \mathbf{x})$ are normal with expected vectors M_i and covariance matrices V_i, then the decision rule becomes

$$\begin{aligned} h(\mathbf{x}) &= -\log(L(\mathbf{x})) \\ &= \tfrac{1}{2}(\mathbf{x}-M_1)^{\mathbf{T}} V_1^{-1}(\mathbf{x}-M_1) - \tfrac{1}{2}(\mathbf{x}-M_2)^{\mathbf{T}} V_2^{-1}(\mathbf{x}-M_2) + \tfrac{1}{2}\log\left(\frac{|V_1|}{|V_2|}\right) \underset{c_2}{\overset{c_1}{\gtrless}} \log\left(\frac{\Pr(c_1)}{\Pr(c_2)}\right) \end{aligned} \tag{3.7}$$

Equation (3.7) illustrates that the decision boundary is expressed quadratically in $\mathbf{x}$. However, when $V_1 = V_2 = V$, the boundary is a linear function of $\mathbf{x}$, expressed as

$$h(\mathbf{x}) = (M_2 - M_1)^{\mathbf{T}} V^{-1}\mathbf{x} + \tfrac{1}{2}\left(M_1^{\mathbf{T}} V^{-1} M_1 - M_2^{\mathbf{T}} V^{-1} M_2\right) \underset{c_2}{\overset{c_1}{\gtrless}} \log\left(\frac{\Pr(c_1)}{\Pr(c_2)}\right), \tag{3.8}$$

where V denotes the system covariance matrix.

The consideration of covariance matrices leads to a special case where $M_i = 0$ and

$$V_i = \begin{bmatrix} 1 & \rho_i & \cdots & \rho_i^{n-1} \\ \rho_i & 1 & & \vdots \\ \vdots & & \ddots & \rho_i \\ \rho_i^{n-1} & \cdots & \rho_i & 1 \end{bmatrix},$$

which can be seen where stationary random processes are sampled temporally to form random vectors. Note that V_i^{-1} and $|V_i|$ are expressed for this form of V_i as

$$V_i^{-1} = \frac{1}{1-\rho_i^2} = \begin{bmatrix} 1 & -\rho_i & 0 & \cdots & 0 \\ -\rho_i & 1+\rho_i^2 & -\rho_i & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -\rho_i & 1+\rho_i^2 & -\rho_i \\ 0 & \cdots & 0 & -\rho_i & 1 \end{bmatrix} \quad \text{and} \quad |V_i| = \left(1-\rho_i^2\right)^{n-1}.$$

This allows us to express the quadratic form of Equation (3.7) as

$$\left[\frac{1+\rho_1^2}{1-\rho_1^2}-\frac{1+\rho_2^2}{1-\rho_2^2}\right]\sum_{i=1}^{n}x_i^2-\left[\frac{\rho_1^2}{1-\rho_1^2}-\frac{\rho_2^2}{1-\rho_2^2}\right]\left(x_1^2+x_n^2\right)$$
$$-\left[\frac{2\rho_1}{1-\rho_1^2}-\frac{2\rho_2}{1-\rho_2^2}\right]\sum_{i=1}^{n}x_i\,x_{i+1}+(n-1)\log\left(\frac{1-\rho_1^2}{1-\rho_2^2}\right)\overset{c_1}{\underset{c_2}{\gtrless}}\log\left(\frac{\Pr(c_1)}{\Pr(c_2)}\right) \tag{3.9}$$

where the second term portrays the boundary effect of finite-length observation of random processes, which diminishes with increasing n. Ignoring the second and fourth terms, and setting $\log(\Pr(c_1)) / \log(\Pr(c_2)) = 0$ [e.g., $\Pr(c_1) = \Pr(c_2)$], one can express the decision rule as

$$\left(\sum x_i x_{i+1}\right)/\sum x_i^2 \gtrless T, \text{ where T denotes a threshold.}$$

In other words, classification occurs by thresholding an estimate of the correlation coefficient. This is reasonable, since $v_1 \neq v_2$ is (in this case) the only difference between c_1 and c_2.

Another interesting well-known case occurs when the pattern vectors are mutually independent and exponentially distributed, which implies that

$$p(c_i \mid \mathbf{x}) = \prod_{j=1}^{n}\frac{1}{a_{ij}}e^{-\frac{x_j}{a_{ij}}s(x_j)}, \tag{3.10}$$

where a_{ij} denotes the exponential distribution parameter for x_j and c_i, with s denoting the step function. As a result, $h(\mathbf{x})$ in Section 3.1.1 can be rewritten as

$$h(\mathbf{x}) = \sum_{j=1}^{n}\left[\frac{1}{a_{1j}}-\frac{1}{a_{2j}}\right]x_j + \sum_{j=1}^{n}\log\left[\frac{a_{1j}}{a_{2j}}\right]. \tag{3.11}$$

In this case, the Bayes decision rule becomes a linear function of the terms x_j.

3.2. Overview of Classifier Performance Analysis

In order to determine whether or not a classifier is performing correctly, we need to take into account the concepts of *cost* and *risk*. For example, in ATR practice, one might prefer to weight different target or non-target objects according to their importance. For example, the cost associated with a classification as *tank* is likely to be of different kind and degree than the object type *schoolbus*. In the two-class problem described herein, the misclassification of a non-target object (e.g., a schoolbus) as a target (e.g., a tank) is likely to be significantly more damaging than the converse. Accordinagly, we present the following brief discussion of cost weighting in Bayesian decision paradigms.

If C_{ij} denotes the cost of deciding that a feature vector derived from an observation (e.g., an AOI in an image) is a member of class c_i (written as $\mathbf{x} \in c_i$) when $\mathbf{x} \in c_j$ is the correct classification, then the *conditional cost of deciding* $\mathbf{x} \in c_i$, which we denote as $r_i(\mathbf{x})$, is given by

$$r_i(\mathbf{x}) = C_{i1}\cdot\Pr(c_1 \mid \mathbf{x}) + C_{i2}\cdot\Pr(c_2 \mid \mathbf{x})\,. \tag{3.12}$$

The decision rule and its resulting *conditional cost* $r(\mathbf{x})$ given $\mathbf{x}$ are expressed as

$$r_1(\mathbf{x}) \underset{c_2}{\overset{c_1}{\gtrless}} r_2(\mathbf{x}) \quad \text{and} \quad r(\mathbf{x}) = \min\left(r_1(\mathbf{x}), r_2(\mathbf{x})\right). \qquad (3.13)$$

The total cost r of this decision is given by the expected value of $r(\mathbf{x})$:

$$r = E\left(\left|r(\mathbf{x})\right|\right) = \int \min\left(r_1(\mathbf{x}), r_2(\mathbf{x})\right) \cdot p(\mathbf{x})\, d\mathbf{x}$$

$$= \int_{L_1} \left(C_{11} \Pr(c_1) p(c_1 \mid \mathbf{x}) + C_{12} \Pr(c_2) p(c_2 \mid \mathbf{x})\right) d\mathbf{x} + \int_{L_2} \left(C_{21} \Pr(c_1) p(c_1 \mid \mathbf{x}) + C_{22} \Pr(c_2) p(c_2 \mid \mathbf{x})\right)$$

where L_1 and L_2 are determined by the decision rule in Equation (3.13).
The decision boundary that minimizes r in the preceding equation can be calculated as follows. First, we rewrite the preceding equation as a function of L_1, by replacing

$$\int_{L_2} p(c_i \mid \mathbf{x}) d\mathbf{x}$$

with

$$1 - \int_{L_1} p(c_i \mid \mathbf{x}) d\mathbf{x}\ ,$$

which can be done since L_1 and L_2 are disjoint but cover the entire domain. We thus obtain

$$r = \left(C_{21} \Pr(c_1) + c_{22} \Pr(c_2)\right) + \int_{L_1} \left[(C_{11} - C_{21}) \Pr(c_1) p(c_1 \mid \mathbf{x}) + (C_{12} - C_{22}) \Pr(c_2) p(c_2 \mid \mathbf{x})\right] d\mathbf{x} \qquad (3.14)$$

By appropriate choice of L_1 (where possible), r can be minimized. If, for a given $\mathbf{x}$, the integrand of Equation (3.14) is negative, then r can be decreased by assigning $\mathbf{x}$ to L_1. If the integrand is positive, it is possible to decrease r by assigning $\mathbf{x}$ to L_2. Thus, the *minimum-cost decision rule* assigns to L_1 the values of $\mathbf{x}$ for which the integrand of Equation (3.14) is negative. This can be summarized in the following inequality:

$$(C_{12} - C_{22}) \Pr(c_2) p(c_2 \mid \mathbf{x}) \underset{c_2}{\overset{c_1}{\gtrless}} (C_{21} - C_{11}) \Pr(c_1) p(c_1 \mid \mathbf{x}) \qquad (3.15)$$

which can be expressed ratiometrically as

$$\frac{p(c_1 \mid \mathbf{x})}{p(c_2 \mid \mathbf{x})} \underset{c_2}{\overset{c_1}{\gtrless}} \frac{(C_{12} - C_{22}) \Pr(c_2)}{(C_{21} - C_{11}) \Pr(c_1)}\ . \qquad (3.16)$$

This is typically called the *Bayes test for minimum cost*. By comparing Equation (3.16) with Equation (3.6), it can be readily seen that this test is a likelihood ratio test with a threshold that is different from the one used in Equation (3.6). In particular, the selection of cost functions is equivalent to changing the *a priori* probabilities $\Pr(c_i)$. It is interesting to note that Equation (3.16) is identical to Equation (3.6) when $C_{21} - C_{11} = C_{12} - C_{22}$, which is called a *symmetrical cost function*. Here, the cost becomes the probability of error, and the test inherent in Equation (3.16) minimizes this probability of error. In a two-class problem, when a wrong decision for one class is more critical than for the other class, an asymmetrical cost function must be employed.

3.3. Refinement of Bayesian Classifier Output

Machine learning (ML) has become an important part of research in artificial intelligence, image and signal processing, and database research. In statistical pattern recognition, one applies probability theory and decision theory to a statistical model of the input to obtain a classification algorithm. This is opposed to using training data to select among different algorithms, or to employing common-sense heuristics to design an algorithm. In classification, one assigns a class identifier to a measurement, or equivalently, identifies the probabilistic source of a measurement. Here, one needs only the conditional model of the class variable given the measurement, which can be obtained from a conditional model or learned directly. The former approach is generative, since it models the measurements in each class, but requires more work, can exploit more a priori knowledge, needs less data, is more modular, and can handle missing or corrupted data. Applicable techniques include mixture models and hidden Markov models. The latter approach (direct learning) is discriminative since it focuses only on distinguishing between classes. This can be more efficient in runtime mode (i.e., pattern classification subsequent to training) and requires fewer modeling assumptions. Typical techniques include logistic regression, generalized linear classifiers, and nearest-neighbor classification. Based on our research, we prefer direct learning due to the previously-listed advantages, as well as its flexibility and reduced reliance on a priori models (which are often brittle in ATR practice).

In Bayesian model selection, one chooses the model that assigns the highest probability to the data after all parameters have been removed (e.g., via integration). In contrast, maximum likelihood estimates parameter values that maximize the classifier's likelihood function. This ignores the prior distribution and thus is inconsistent with Bayesian theory, but works well in practice. Alternatively, maximum a posteriori (MAP) heuristics seek parameter values that maximize a parameter's posterior density. This implies the Bayesian assumption of a prior distribution over the parameter. Although MAP estimation has generally high accuracy, it tends to predict future data less accurately than maximum likelihood techniques. A better Bayesian method for parameter estimation is predictive estimation. Also, MAP is sensitive to reparameterization, which is central to adaptivity in the proposed GASP scheme of ML.

An important implementational technique in ML is the induction of decision trees, which tend to be more comprehensible to humans than other learning models such as neural networks [Gro99]. Decision trees efficiently solve the ATR classification problem (i.e., predicting a class identity based on target/feature attributes), since classification requires only a linear walk down the tree. Bagging and Boosting [Die98] are efficient techniques for improving classification accuracy of a single decision tree, by generating multiple learning methods, then voting on their output.

Grossman and Williams [Gro99] improved the Bagging algorithm introduced by Breiman [Bre94], based on the ID3 decision tree algorithm with chi-squared pre-pruning. Learning methods are weighted, where weights are based on the classification accuracy of each learner on samples not selected during the bootstrapping process, which generates random training sets for each learner from the training data. In contrast, Hashem [Has94,Tum96] uses least-squares regression to discover weights that maximize classification accuracy. Additionally, Indurkhya and Weiss tested alternate decision trees based on pruning by statistical significance levels, on a subset of the training data to find the best-performance tree [Ind98]. They also employed a voting mechanism based on this tree.

The work of the preceding authors was based on the assumption that testing on a verification set estimates the future performance of a decision tree. Grossman and Williams generate a sufficient number of multiple trees by choosing bootstrap sets from the original training data. They use a dynamic weighting function for estimating the goodness of each tree in the ensemble, but do not discard trees in the ensemble that perform below the goodness threshold prior to voting.

Although theory for a posteriori refinement of decision tree based classification has advanced considerably [ScF97,Fri98], there appears to be no easy method for increasing the adaptability of such classifiers or follow-on multi-classifier output disambiguation procedures. For example, classification with Dempster-Schaefer theory has been based on assumptions of stationary Markov models [Cla91]. However, practical target recognition problems involve nonstationary input, to which successful, robust classifiers must adapt. As mentioned in previous sections, a key disadvantage of Bayesian approaches is the requirement of significant a priori knowledge, a statistical model of the input that must incorporate key features that are themselves input to the pattern classifier(s), and decision structures that must be adapted in a supervised method – a globally optimal construction method is presently unclear. Additionally, the noise tolerance of Bayesian classifiers has only been recently reported for experimental ATR scenarios [Gre00], and Bayesian classifiers do not necessarily adapt well to nonergodic inputs – for example, the vast majority of reports in the literature describe designs based on ergodic Markov processes [Wan00,Got98,Bor96].

Traditionally, the adaptability of pattern recognizers is conceptualized as having two modalities, which we have previously called *build-time* and *runtime.* During build time, the classifier can be adapted offline (i.e., trained) on a static dataset. Input changes can be tracked by maintaining a history of input statistics. During runtime, the classifiers and their associated output disambiguation or postprocessing routine(s) are applied to input data in the usual manner. To remedy well-known problems of lack of adaptivity associated with traditional rule-based classifiers, we have employed a novel multi-classifier approach. In particular, we have assisted FTI in employing a morphological NN to process the TNE agreement map columns, which correspond to preliminary classifier output, as discussed in Section 2. It has been shown that MNNs perform superiorly to classical NNs [Rit97,98]. MNN technology is further discussed in Section 5.

In a possible Phase 2 effort, we propose to assist FTI in further extending the basic concept of Grossman and Williams' technique to include adaptive classifier refinement techniques based on UF's MNNs and FTI's TNE paradigm. As in the current study, the fusion and disambiguation of multiple classifier outputs can be modulated by classifier performance data determined by standard ATR performance analysis metrics (e.g., Pd and Rfa). This closed-loop modulation of the classifier refinement process supports adaptation to performance challenges such as input nonergodicities via a backpropagation algorithm that closely resembles established learning procedures that have been successfully applied to classical NNs. Observe that accurate, nontrivial tree- or rule-based classifiers that continuously adapt to fluctuations in input statistics have yet to be reported in the literature. Similar in overview to the implementation of traditional classifiers, MNN, and TNE-based classifiers are illustrated notionally in Figure 3.1. However, the combination of adaptability and accuracy inherent in the joint MNN/TNE classifier represents a significant development in pattern recognition technology. The concept of using

MNNs to drive adaptation of TNE-based pattern recognition represents an important advance over traditional open-loop ATR systems.

A second use of MNNs would be in supporting target cluster analysis and optimization in the template database. In practice, targets tend to exhibit similarity at slightly different poses, due to tangent projection geometry. That is, a tank rotated out-of-plane by five degrees will appear visually similar to the same tank rotated out-of-plane by 10 degrees. A preliminary estimate of target similarity can be arrived at by application of a Euclidean distance function to the M target exemplars, each of which has K pixels, thereby incurring work of $\mathbf{O}(KM^2)$ multiplications and $\mathbf{O}(M^2)$ square roots. Alternatively, an MNN could be configured to examine the target exemplars, taking into account factors such as background mean, spatial and grayscale variance, or feature orientation. Each of these measures could be determined by an MNN, and could provide useful information about target similarity beyond the distance functions typically (but not necessarily) employed by TNE.

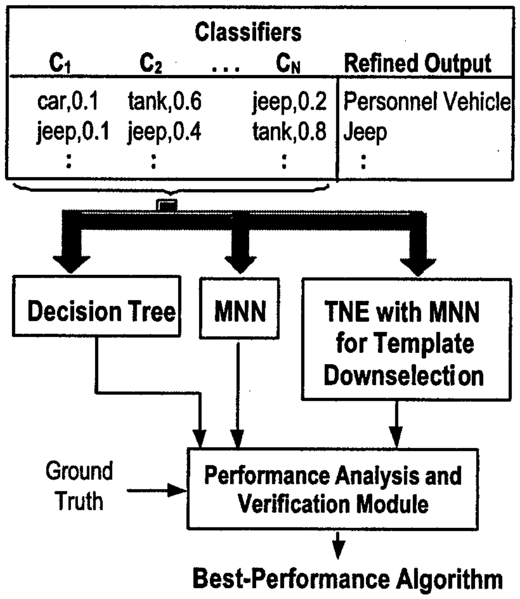


Figure 3.1. Experimental configuration for competitive performance analysis of classifier refinement algorithms.

3.4. Utility of Bayesian Classification for ATR

The preceding discussion highlights several features of Bayesian decision paradigms:

1. *Coverage must be complete* – that is, all outcomes must be known. This means that Bayesian classifiers, as presented in the classical Bayesian theory of this section, cannot accurately interpolate or extrapolate over missing data.
2. *Probabilities must be computed accurately*: if a given outcome is unknown or guessed at, then the *a priori* probability will be erroneous, which will influence the computation of *a posteriori* probabilities and, therefore, the associated likelihood ratio(s). This means that Bayesian cost minimization will, in the affected instances, be erroneous and thus, nonoptimal. In practice, therefore, the presence of erroneous probabilities mean that the cost function cannot be minimized.

3. *A given classifier comprised of a fixed system of Bayesian rules cannot readily be re-parameterized* to include additional classes, rules, or instances of input patterns. If such additional constraints are introduced, then the prior knowledge changes, which in each nontrivial case modifies the *a posteriori* probabilities. As in item 2, above, this affects the likelihood ratios and cost function minimization.

The preceding features represent significant limitations of classical Bayesian pattern classifiers.

Reasoning under uncertainty and incomplete data are, however, supported by FTI's TNE paradigm, which is discussed in Section 4. Further support for such realistic constraints is provided by neural network-based pattern classifiers, in particular, UF's Morphological Neural Networks, which is discussed in Section 5. In particular, neural nets have well-known capabilities of accurate pattern classification given partial or noise-corrupted inputs, and are extensible in the sense that they can learn new patterns and classification rules. Thus, we next discuss these paradigms, then overview their joint application to ATR in Section 6. A summary of advantages and disadvantages of each approach is given in Section 7.

3.5. References

[Bor96] Borkar, V.S., and S.K. Mitter. "Stochastic processes that generate polygonal and related random fields", *IEEE Transactions on Information Theory* **42**(2):606-617 (1996).

[Bre94] Breiman, L. "Bagging Predictors", Technical Report 421, University of California/Berkeley (1994).

[Cla91] Class, F., A. Kaltenmeier, and P. Regel. "Soft-decision vector quantization based on the Dempster/Shafer theory", *Proceedings IEEE 1991 International Conference on Acoustics, Speech and Signal Processing* **1**:665-668 (1991).

[Die98] Dietterich, T.G. "Machine-Learning Research: Four Current Directions", *AI Magazine* **18**(4):97-136 (1998).

[Fri98] Friedman, J., T. Hastie, and R. Tibshirani. "Additive logistic regression: A statistical view of boosting", Dept. of Statistics, Stanford University Technical Report (1998).

[Got98] Gotoh, M, T. Matsushima, and S. Hirasawa. "Generalization of B.S. Clarke and A.R. Barron's asymptotics of Bayes codes for FSMX sources", *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* **E81-A**(10):2123-2132 (1998).

[Gro99] Grossman, D. and T. Williams. "Machine Learning Ensembles: An Empirical Study and Novel Approach", Project Report (1999), available from http://citeseer.nj.nec.com/391913.html (31 Mar 2002).

[Has94] Hashem, S., B. Schmeiser, and Y. Yih. "Optimal linear combinations of neural networks: an overview", *Proceedings of the IEEE International Conference on Neural Networks* **3**:1507-1512 (1994).

[Ind98] Indurkhya, N. and S.M. Weiss. "Estimating Performance Gains for Voted Decision Trees" IBM Research Division Technical Report RC-21199, in Intelligent Data Analysis (1998).

[Rit97] Ritter, G.X. and P. Sussner. "Associative memories based on lattice algebra", *Proceedings of the 1997 IEEE International Conference on Systems, Man, and Cybernetics* **4**:3570-3575 (1997).

[Rit98] Ritter, G.X., P. Sussner, and J.L. Diaz-de-Leon. "Morphological associative memories", *IEEE Transactions on Neural Networks* **9**(2):281-293 (1998).

[ScF97] Schapire, R.E., Y. Freund, P. Bartlett, and W.S. Lee. "Boosting the margin: a new explanation for the effectiveness of voting methods", *Proc. 14th International Conference on Machine Learning*, pp. 322-330 (1997).

[Tum96] Tumer, K. and J. Ghosh. "Analysis of decision boundaries in linearly combined neural classifiers", *Pattern Recognition* **29**(2)341-348 (1996).

[Wan00] Wang, X., and R. Chen. "Adaptive Bayesian multiuser detection for synchronous CDMA with Gaussian and impulsive noise", *IEEE Transactions on Signal Processing* **48**(7):2013-2028 (2000).

4.0 TNE Pattern Recognition Paradigm

Tabular Nearest Neighbor Encoding (TNE) is an advanced pattern recognition paradigm that employs mapping operators to construct an intermediate data array called the *agreement map* (AM) from prior knowledge applied to input data. The AM is then processed (for example, via thresholding) to yield a Boolean data structure that can be subjected to bitwise *or*, *and*, as well as *xor* operations, followed by row or column summation to yield classification scores. Because TNE is based on an efficient projection technique that performs preliminary detection of candidate pattern features in hyperspace, and because this projection is implemented in terms of I/O operations only, TNE is significantly more efficient than template matching techniques based on arithmetic operations. Section 4.1 presents basic theory of TNE, while Section 4.2 compares TNE with previous and related research. Section 4.3 contains several applicative examples and complexity analysis. Section 4.4 presents a heuristic overview of the paradigm applied to multi-source data fusion.

4.1. Mathematical Description of TNE

We begin with an overview of TNE concepts, then progress to the TNE algorithm. Let an N-pixel input (e.g., a $\sqrt{N}$ x $\sqrt{N}$ -pixel source image $\mathbf{a} \in \mathbf{Z}_m^N$) be mathematically subdivided into K-pixel sampling blocks, of which there are N/K such blocks or *test vectors*. Each vector $\mathbf{b} \in \mathbf{Z}_m^K$ has K *vector components*. Sampling blocks can be rectangular or noncontiguous, and the vector components can be groups of pixels, streaming audio or video subsamples, etc.

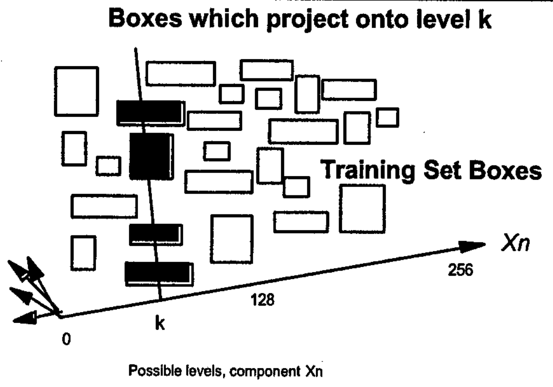
Further assume that there exists a database **c** of M reference patterns, each having K pixels and being configured in the same manner as **b**. Thus, $\mathbf{c}(i) \in \mathbf{Z}_m^K,\ 1 \le i \le M$. For example, if **b** is comprised of square blocks of $\sqrt{K}$ x $\sqrt{K}$ pixels, then each *exemplar* **b**(*i*) in **c** has $\sqrt{K}$ x $\sqrt{K}$ pixels.

In traditional pattern recognition applications, a technique called *exhaustive template matching* compares each K-pixel source block **b** with each K-pixel exemplar in **c**, requiring work of at least NMK pixels per source image, if all sampling block phase shifts are accounted for. If one applies such operations to large images with many reference patterns (e.g., N = 10^6, M = 10^5, K = 10^2), then prohibitive work (e.g., NMK = 10^{13} comparison operations) is incurred.

In contrast, TNE precomputes a projection D that stores all possible outcomes resulting from application of the M-exemplar pattern database to each possible source block configuration. For example, if the source value set is $\mathbf{Z}_m$, then there are m^K possible configurations of each source block. Because TNE uses I/O operations only to implement p, in a processor having fast I/O, this approach is significantly more efficient than using comparison operations. We next present theory that describes the TNE algorithm, in particular, the projection D that supports TNE's implementational efficiency.

4.1.1. Assumptions. Let an image $\mathbf{a} \in \mathbf{F}^{\mathbf{X}}$ be subdivided by an indexing function h to yield a collection of K-pixel sampling blocks $A = \{\mathbf{b}(\mathbf{y}) : \mathbf{y} \in \mathbf{Y}\}$, where $\mathbf{Y} \subset \mathbf{X}$. Let a codebook **c** be formed from A, such that **c** contains M K-pixel exemplars, each of which represent a cluster C_i, where i = 1..Q. Let a feature space representation F have axes B_1, B_2, ..., B_j, ..., B_P, to which are projected each of the clusters C_i, thereby producing a collection of intervals denoted by

$$I = \{I_{i,j} \in \mathbf{R}^2 : 1 \le i \le M \text{ and } 1 \le j \le P\}.$$


Boxes which project onto level k
Training Set Boxes
Xn
256
128
k
0
Possible levels, component Xn

Level K boxes: **2, 5, 6, 9, 14, ...**

Level K Binary vector: **01001100100001...**

Figure 4.1: Derivation of binary pointer vectors for each quantization level along each dimension of a highly-dimensional feature space..

This projection process is illustrated notionally in Figure 4.1, in which the implementational term *binary pointer vector* corresponds to a projection of multiple pattern clusters.

4.1.2. Algorithm. Let a sampling block **b**(**y**) be represented by a point or region **p** in a P-dimensional feature space F. Let **p** be projected to axes B_j, j = 1..P, to yield a collection of intervals denoted by

$$J = \{J_{i,j} \in \mathbf{R}^2 : 1 \le i \le M \text{ and } 1 \le j \le P\}.$$

Although J_j is one-dimensional when **p** is a point (the simplest case), we employ the more involved assumption that $J_{i,j}$ is two-dimensional, for purposes of generality.

Step 1. Let I and J be processed by an operation that compares the extent of $J_{i,j}$ with the extent of $I_{i,j}$, such that a PxM-element bitmap **d** is formed, as follows:

$$\mathbf{d}(i, j) = \begin{cases} 1 & \text{if } p_1(J_{i,j}) \ge p_1(I_{i,j}) \text{ and } p_2(J_{i,j}) \le p_2(I_{i,j}) \\ 0 & \text{otherwise} \end{cases},$$

where p_k denotes projection to the kth coordinate. This process is also illustrated notionally in Figure 4.1.

Step 2. As an example of pattern recognition, sum **d** rowwise, and subtract P as to yield scores $\mathbf{g}(i) = P - \Sigma\mathbf{d}_i$, where $\mathbf{d}_i$ denotes the i-th row of **d**. The resultant scores equal the Hamming distances between **p** and each exemplar **c**(i) represented by cluster C_i. Associated AM generation and manipulation steps are shown notionally in Figure 4.2.

Step 3. The best-match codebook exemplars are given by **c**(*domain*(*min*(**g**))); the *choice* function can be used to select one best-match exemplar randomly from these exemplars. An example matching result is shown schematically in Figure 4.3.

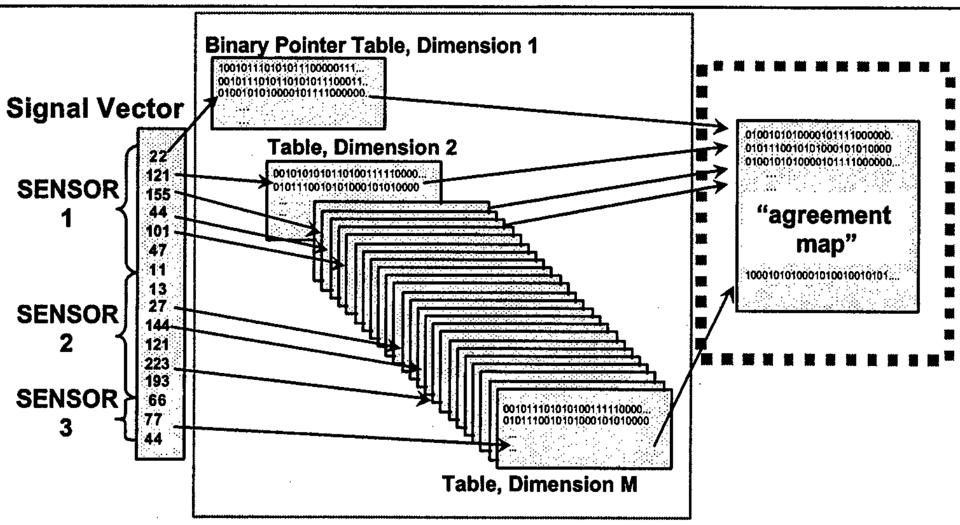


Figure 4.2. Specification of the virtual agreement map via pointers derived from sampled vector components. In this notional example, the AM implements the fusion of three distinct sensors or sensor data types.

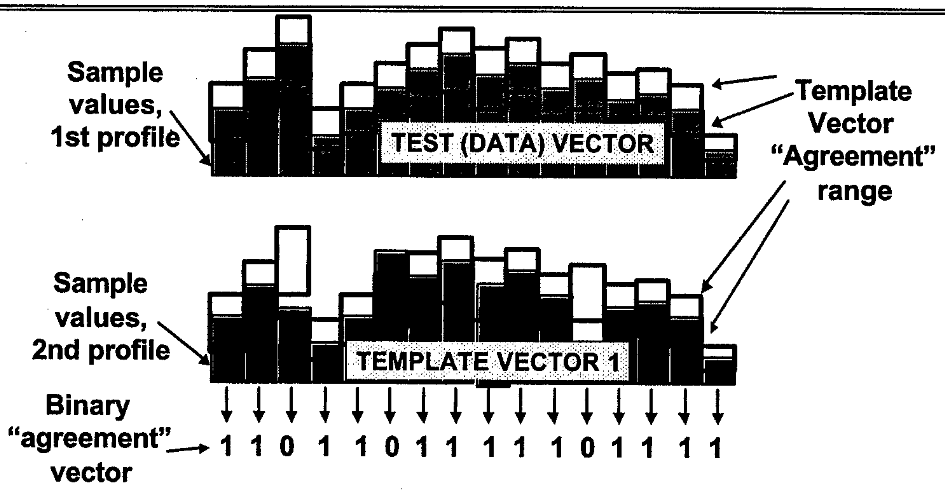

Sample values, 1st profile
TEST (DATA) VECTOR
Template Vector "Agreement" range
Sample values, 2nd profile
TEMPLATE VECTOR 1
Binary "agreement" vector
1 1 0 1 1 0 1 1 1 1 1 0 1 1 1 1

Figure 4.3: TNE pattern recognition process combines the source vector with the training set vector in the context of the agreement map. In particular, TNE derives a binary agreement vector wherein a 1 signifies that the associated components agree within a prespecified tolerance, and a 0 indicates no match.

4.1.3. Complexity. Assuming that the codebook cluster projections are precomputed, as mentioned previously, the projection of **p** to the axes of F requires **O**(P) arithmetic and transcendental operations per source block, for example, P sine operations and 2P additions. Comparison of I and J requires 2PM comparisons per source block, with P(M+1) additions required to produce **g**. Similarly, M comparisons are required to find the best-match exemplar in **c**. Hence, the work required by TNE codebook search over **a** is given by:

$$W_{\text{TNE}} = \text{N(M(2P + 1) comparisons} + \text{P(M + 3) additions} + \text{P transcendentals)} .$$

4.1.4. Observation. In previous research [Key99], we have presented TNE as a means for implementing fast codebook search over imagery compressed by vector quantization (VQ). However, the TNE algorithm does not necessarily compress an image, but primarily provides an efficient means for codebook search. TNE can be configured to compress an image by indexing each sampling block according to the spatial configuration of its grayscale values. That is, a given pixel (**x**,**a**(**x**)) of **a** provides both spatial and grayscale information to a map $D: \mathbf{X} \times \mathbf{R} \rightarrow \mathbf{G}$, where **R** denotes *domain*(**a**) and **G** is an indexed set of pointers to M-bit Boolean vectors stored in database D. Each vector represents one of the K pixels of a given sampling block **b**. In the resulting MxK-pixel array **d**, which is called the *agreement map,* the j-th column represents a bit vector of binary matching scores between (1) value **b**(**x**) at position **x** of *domain*(**b**) indexed by j, and (2) all exemplar values **c**(i)(**x**), where i =1..M. The exemplar that best matches **b** is given by

$$\mathbf{c}(choice(domain(min(\text{K} - \Sigma\mathbf{d}_i)))) ,$$

where $\mathbf{d}_i$ denotes the i-th row of **d**.

4.1.5. Remark. It is easily verified that precomputation of D is the burdensome step in the TNE algorithm, which can be compared to the overhead of codebook construction. For example, if each sampling block has K pixels each having m greylevels, then m^K block configurations are possible. Comparison of these configurations with the M codebook exemplars yields a total cost of W = **O**(KMm^K) comparison operations. Given small values of K = 64, M = 256, and m = 256, it is easily verified that W is prohibitively large. Hence, it is reasonable to determine the subset S of the m^K block configurations that occurs in a given training set. Given S, W can be reduced to **O**(KM·|S|) comparison operations. For example, if |S| = 10^5 and the proportionality constant in the complexity estimate of W is set to unity for purposes of simplicity, then W = 256^3x10^5 = 1.67 x 10^9 comparison operations. Derivation of the working set S is a topic of current research, which we plan to emphasize in a possible Phase-II effort.

4.2. Previous Work

4.2.1. Observation. As noted previously, a problem with VQ codebook search has been achieving log(M) time complexity. Efficient codebook search algorithms that have been published in the open literature include:

1) *n-ary tree search*, where each exemplar is rendered in progressive n-fold resolution reduction, to facilitate multiresolution representation per source block [Xu98]
2) *Row and column decimation* schemes, whereby a sampling block is characterized in terms of a signature derived from each row or column, to facilitate comparison of fewer features [ElS99];
3) *Statistical search* techniques, whereby each codebook exemplar and source block are represented by parameters such as mean, standard deviation, skewness, or kurtosis [Pog93,Sch97];

4) *Subpattern characterization*, in which each codebook and exemplar is subdivided into partitions, each of which correspond to a small pattern, thus producing a hierarchy of codebooks [Che99]; and
5) *Block transformations* such as the discrete cosine transform employed in JPEG, to reduce each encoding block to a vector of coefficients that can be quantized and downselected [Mey98].

The common feature of each search method is the reduction of an exemplar or source block to an approximate representation, which can be thought of as a *signature* or *feature vector*.

4.2.2. Remark. Unfortunately, hierarchical, row/column, statistical, subpattern, and transformation based representational schemes are n-to-1 mappings and are thus lossy transforms. As a result, the representation of a given source block may not be sufficiently precise to avoid confusion between exemplars. Therefore, one usually implements a multi-level matching scheme to compensate for this deficiency. An illustrative example follows.

4.2.3. Algorithm. In multi-level matching, given a source block **b**, one performs the following steps:

Step 1. Reduce **b** to an approximate representation, such as a block mean or simplified zero-crossing pattern derived by thresholding **b** about its mean.

Step 2. Compare the reduced block to a similar, reduced representation of each codebook exemplar. Alternatively, the block representation can be structured to point to a block indices in a list of N_e probable best-match exemplars.

Step 3. If the latter alternative in Step 2 is employed, then each of the N_e exemplars would be compared to **b**, yielding a best match according to a prespecified matching criterion.

4.2.4. Complexity. Reduction of the source block to a parametric representation typically requires f_sK sampling operations, where the fraction f_s denotes a subsampling ratio. For example, if $f_s = 0.4$, then two out of every five source pixels are sampled. Table 4.1 lists the complexity of various reduction operations, including n-ary tree encoding. If a given representational system uses N_p parameters and a reduced source block representation, then comparison of N_e exemplars requires at least N_pN_e operations. The cost of different comparison methods is listed in Table 4.2. Observe that threshold techniques listed in Table 4.2 generally compare a source block value v to determine if v is in a prespecified interval, which can be space-variant. The Sum Threshold (Product Threshold) comparison sums (resp. takes the product of) the Boolean values that represent the partial thresholded matches.

Table 4.1. Work W_r incurred by a reduction operation applied to a K-pixel source block with sampling factor f_s.

Block Operation	Additions	Multiplications	Comparisons	Roots
Mean μ; work = W_μ	$\lceil f_sK \rceil - 1$	1	0	0
Std. Deviation σ; W_σ	$\lceil 2f_sK \rceil - 1$	$\lceil f_sK \rceil + 1$	0	1
kth Central Moment m_k	$\lceil kf_sK \rceil - 1$	$\lceil (k-1)f_sK \rceil + 1$	0	1
n-ary Tree Encoding	$\sum_{i=0}^{n} \frac{\lceil f_sK \rceil}{n^i} \cdot \frac{K}{n^{i+1}}$	$\sum_{i=0}^{n} \frac{K}{n^{i+1}}$	0	0

Table 4.2. Work W_m incurred by a matching operation applied to an N_p-parameter source block representation.

Block Operation	Additions	Multiplications	Comparisons	Roots
Sum of Deviations	$N_p - 1$	0	0	0
Mean Deviation	$N_p - 1$	1	0	0
Sum-Squared Deviation	$N_p - 1$	N_p	0	0
RMS Deviation	$2N_p - 1$	N_p	0	1
Normalized Autocorrelation	0	$2N_p + 1$	0	0
Normalized Cross-correlation	0	$3N_p + 1$	0	0
Sum of Thresholds	$N_p - 1$	0	$2N_p$	0
Product of Thresholds	0	N_p	$2N_p$	0

The mean work required for block encoded compression via these types of reductionist codebook search techniques can thus be expressed in terms of blocksize K, mean number of exemplars N_e retrieved during primary search, and the work W_r (W_m) required by a block reduction (matching) operation, as follows:

$$W_{tot} = W_r(f_s, K) + N_e \cdot W_m(N_p) .$$

Note that W_m is not weighted by f_s (although the work estimates in Table 4.1 could be so weighted), since computation over the reduced representation is already efficient.

4.2.5. TNE-like Pattern Matching Paradigms. Ramasubramaniam and Paliwal [Ram99] recently published a search paradigm similar to TNE that is amenable to joint implementation with vector quantization. Two techniques are investigated (*box-search* and *cell-partition search*), which are based on Voronoi projections. The cell-partition method uses an intersection-count procedure for obtaining a set of candidate vectors, whereas the box-search algorithm uses a bounding-box interpretation of the projection information. Since it does not maintain an intersection count, *box-search* requires significantly less storage. Both techniques produce an optimum set of codevectors. The authors also investigated the effect of principal component rotation and found that it reduces the complexity of codebook search based on Voronoi projection, while potentially reducing representational error in the reconstructed (decompressed) data.

4.2.5.1. Box-search. Given codebook **c**, let the Voronoi region V_i associated with a K-dimensional codebook exemplar $\mathbf{c}_i$ contain all points in the space $\mathbf{R}^K$ that are nearer to $\mathbf{c}_i$ than to any other codevector. If a test vector $\mathbf{x} \in V_i$, then $\mathbf{c}_i$ is the nearest neighbor of **x**. The projection $P_{i,j}$ of V_i onto the j-th coordinate axis of $\mathbf{R}^K$ is an interval with upper(R) and lower (L) boundary points

$$P^L_{i,j} = \bigwedge_{\mathbf{x} \in V_i} p_j(\mathbf{x}) \quad \text{and} \quad P^U_{i,j} = \bigvee_{\mathbf{x} \in V_i} p_j(\mathbf{x}),$$

where p_j denotes projection to the j-th coordinate. The preceding bounds represent two hyperplanes normal to the j^{th} coordinate axis. Taken together, the K projections $P_{i,j}$, j = 1..K, define a set B_i of 2K hyperplanes that bound V^i as

$$B_i = \left\{ \mathbf{x} : P^L_{i,j} \le p_j(\mathbf{x}) \le P^U_{i,j}, j = 1..K \right\}.$$

Since each codevector has such a hypercuboid approximation to V_i, we observe that $V_i \subseteq B_i$ implies that a given test vector **x** may have several associated candidate codevectors in the set $C(\mathbf{x}) = \{\mathbf{c}_i : \mathbf{x} \in B_i\}$. Thus, the central task of the *box-search* algorithm is to locate the nearest neighbor of **x** in C(**x**).

This search can be rendered efficient by testing each of the K coordinates of **x** against the bounds of V_i that corresponds to $\mathbf{c}_i \in C(\mathbf{x})$. If the j-th coordinate of **x** falls outside $B_i(j)$, then $\mathbf{c}_i$ is rejected as a candidate codevector. The complexity of this initial search step is thus $\mathbf{O}(K \cdot |C(\mathbf{x})|)$

comparisons. The reduced set C' obtained by initial search of C is then subjected to distance computation, thereby incurring **O**(K) arithmetic operations. This is precisely the method employed in TNE based compression, which uses a Mahalanobis-like distance as the matching metric.

4.2.5.2. Cell-partition search. In the preceding section, N overlapping intervals $P_{i,j}$ generate 2N projection boundaries $(y_{j,1}, y_{j,2}, ..., y_{j,N})$ having a natural ordering $y_{j,1} \leq y_{j,2} \leq ... \leq y_{j,N}$ that partitions the j^{th} coordinate axis into 2N-1 contiguous intervals $I = \{I_j(1), I^j(2), ... , I^j(2N-1)\}$, where $I_j(m) = (y_{j,m}, y_{j,m+1})$. Each interval $I^j(m)$ is associated with a set $S^j(m)$ of indices for V_i whose projection intervals $P_{i,j}$ can overlap with $I^j(m)$.

Further, the 2N-1 contiguous intervals on each of the K coordinate axes partitions $\mathbf{R}^K$ into $(2N-1)^K$ hyperrectangular cells. Given test vector $\mathbf{x} = (x_1, x_2, ... , x_K)$, if $I^j(m_j)$ contains x_j, then $S^j(m_j)$ contains the candidate codevectors that could be nearest neighbors of **x**. The test vector is thus located by examining K intervals $I^j(m_j)$, j = 1..K, to yield the hyperrectangular cell

$$H(\mathbf{x}) = \prod_{j=1}^{K} I^j(m_j) .$$

Thus, the final set of candidate codevectors is given by

$$\underline{S}(\mathbf{x}) = \bigcap_{j=1}^{K} S^j(m_j) .$$

This development suggests to techniques for finding the candidate codevectors C'(**x**). First, one can entabulate the $(2N-1)^K$ cells in a table, from which C' is obtained, here the best-match cell is identified using the preceding equation. Unfortunately, this requires prohibitive storage that increases exponentially as $\mathbf{O}(N^K)$. The second technique precomputes the candidate sets $S^j(m)$ and stores them in K mapping tables. Each table corresponds to one coordinate axis and has 2N-1 rows, one per interval $P_{i,j}$. This requires **O**(K(2N-1)) storage. The final set of candidate codevectors is obtained by an *intersection count* procedure, which requires Klog(2N) + W operations, where $K+N \leq W \leq (K+1)N$. This is similar to the possible TNE matching procedure of summing columns of the agreement map to yield a basis for Hamming distance computation. Such summation can be applied to single columns or to columns grouped via rowwise operations such as Boolean *or*, *and*, or *xor*.

4.2.5.3. Principal component rotation. Ramasubramanian and Paliwal state that the application of principal component rotation to the K-dimensional exemplar space referred to in Sections 4.2.5.1 and 4.2.5.2 results in a consistently lower complexity for *box-search* than the unrotated case. This occurs since the projections of the Voronoi regions obtained with the rotated coordinates significantly reduce the overlap between bounding boxes B_i, thereby reducing the size of set C'. In contrast, the storage and computational overheads of *cell-partition search* are significantly increased, since coordinate rotation increases the average index set size. An additional effect of rotation is increased signal-to-noise ratio (i.e., decreased reconstruction error). Again, this is due to reduced overlap between Voronoi regions. Also, the authors report that rotation groups the SNR closer to that obtained via full search for all codebook sizes. We plan to investigate application of this technique to TNE-based compression in the near future. However, Ramasubramaniam and Paliwal's box-search algorithm does not of itself appear to have advantages over TNE, which generalizes the box-search technique.

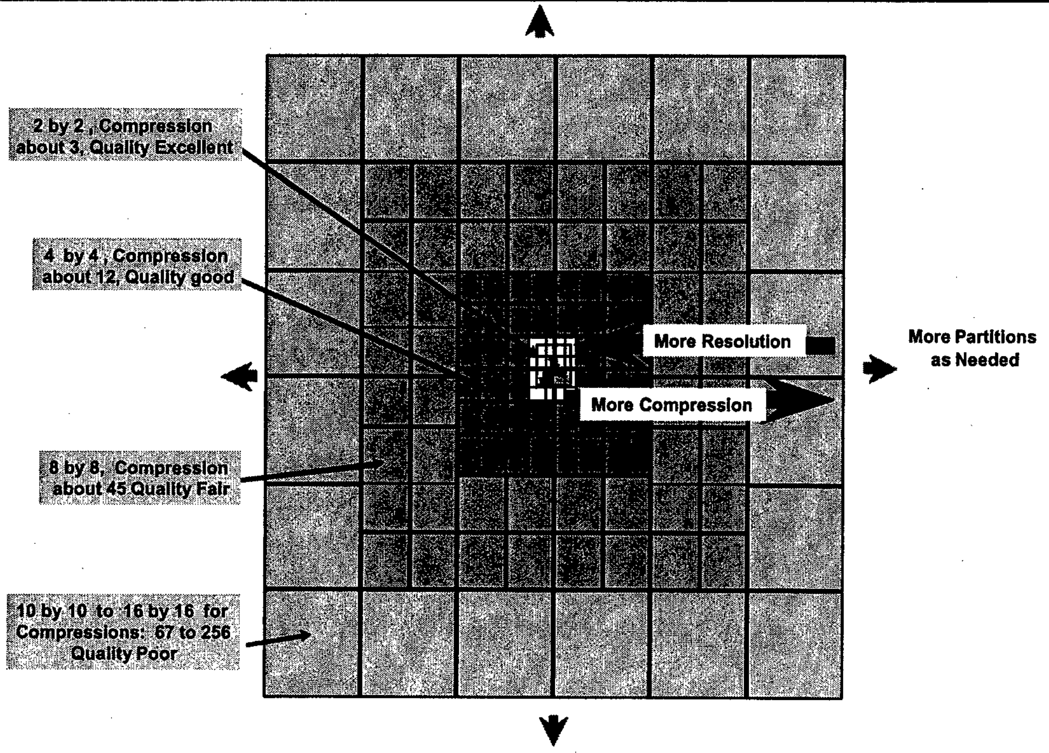

2 by 2, Compression about 3, Quality Excellent
4 by 4, Compression about 12, Quality good
8 by 8, Compression about 45 Quality Fair
10 by 10 to 16 by 16 for Compressions: 67 to 256 Quality Poor
More Resolution
More Compression
More Partitions as Needed

Figure 4.4. Notional diagram of spatial partitioning strategy for TNE/VQ image compression.

4.3. Examples and Analysis

A useful example application of TNE for purposes of initial illustration is image compression, with feature or object recognition over compressed imagery. In this application, TNE can be thought of as involving three steps: (1) image database partitioning, (2) codebook construction, and (3) codebook search, which are described as follows.

4.3.1. Partitioning. As described previously, each source frame or image can be partitioned into K-pixel encoding blocks. Each encoding block can be compressed in two ways. First, the number of source pixels N_{SP} can be reduced to N_{CP}, leading to a *domain compression ratio* $CR_D = N_{SP} / N_{CP}$. Second, the number of bits per source pixel N_{SB} can be reduced to N_{CB} bits per pixel, leading to a *range compression ratio* $CR_R = N_{SB} / N_{CB}$. We have elsewhere shown that the net compression ratio is given by

$$CR = CR_D \cdot CR_R .$$

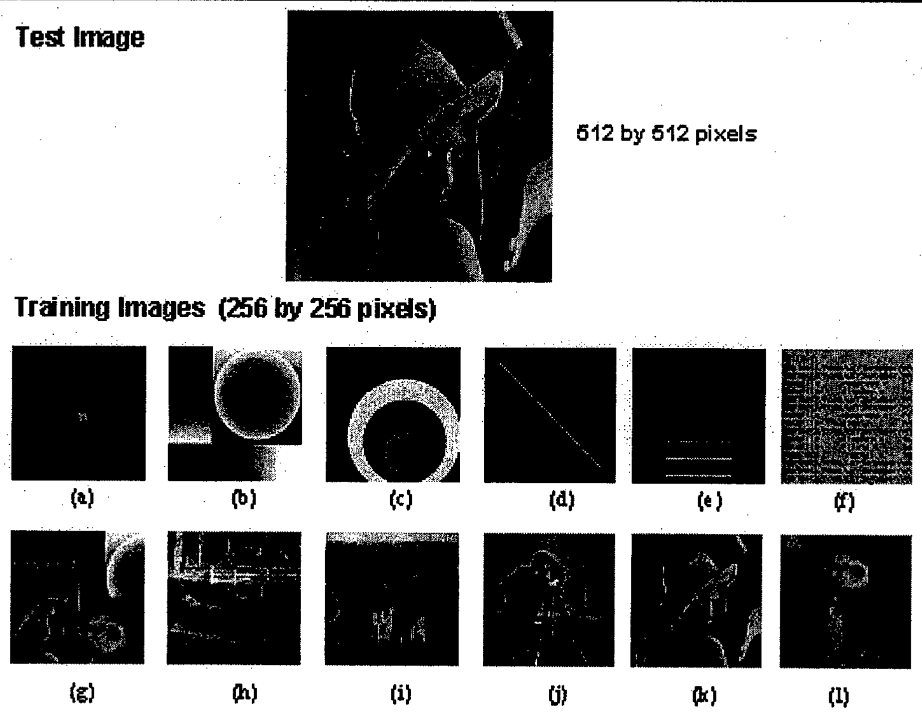


Figure 4.5. Training images (256x256 pixels): (a-e) geometric shapes, (f-g) images with text, (h-j) natural scenes, and (g,j-l) images containing people or animals. Note that *lena* is represented only in g) and k).

Since blocksize directly impacts the domain compression ratio CR_D, this also affects CR. Thus, larger blocks tend to increase the compression ratio, albeit at the expense of visual image quality due to blocking effect. This process is illustrated in the notional diagram of Figure 4.4. This illustration can also depict multiresolution partitioning, which is possible with TNE. In the context of military applications, ATR in conjunction with multiresolution image registration and compression is a topic of our ongoing TNE research, which we propose to further explore as a subtask in a possible Phase-II effort.

4.3.2. Codebook Construction. The codebook employed in the TNE compression/pattern recognition algorithm discussed herein was constructed from the diverse training set shown in Figure 4.5. Note that this training set is not necessarily well suited to the *lena* test image, but has been chosen for its generality. The codebook was constructed using the standard Lloyd-Max algorithm, since codebook construction efficiency is not a consideration in this study.

4.3.3. TNE Codebook Search. Since TNE primarily employs bit operations, execution is efficient at the level of feature vectors. The current prototype of TNE is coded in Microsoft FORTRAN (32-bit architecture). The host machine for timing benchmarks listed herein is based on an Intel Pentium processor running at 133 MHz, with a 256 KB cache. In previous tests on a Pentium II processor running at 266 MHz with a 512 KB cache, the execution times were reduced by a factor of approximately 4.3. In a completed implementation, TNE's execution time could be significantly decreased by coding in C or assembly language. Benchmark times cited below are for encoding (i.e., discovery and assignment of a best-match exemplar index for each source block in an image) as well as decoding (projecting the exemplar indexed by a compressed pixel onto the corresponding source domain). On a Pentium 133 MHz processor, the decoding step required approximately 0.3 seconds for a 512x512-pixel image, for exemplar size ranging from 4x4 pixels to 16x16 pixels. Further details of execution times as a function of exemplar and codebook size are given in [Key99], where we showed that TNE offers a speedup, as opposed to search based on a Euclidean distance matching criterion, ranging from 52.8:1 to 129.8:1, which is a typical performance gain for the type of imagery and computer employed in these tests. As M increases, the resultant speedup tends to increase, since the work required for TNE is **O**(log M), while Euclidean-based search incurs at best **O**(M) work.

We next consider ATR operations over TNE encoded imagery.

4.3.4. Example Application: TNE-based ATR. TNE can be applied to target recognition by constructing an agreement map based on a database of target templates. As described in Section 3.1, the TNE encoding of each target template comprises a column of the agreement map selected from the template database according to source pixel greylevel and location within an encoding block, as shown in Figure 4.6. Bit operations on the agreement map columns, or between groups of columns in the agreement map, comprise the analogues of an image template correlation operation.

In practice, TNE-based target classification is an analogue of a template matching operation over the range space of a TNE-encoded image. In TNE, target templates are encoded in the agreement map columns, and matching is performed over a TNE encoding of each source block. This is exemplified in Figure 4.7, which demonstrates the superiority of TNE's matching technique versus Euclidean distance based classification.

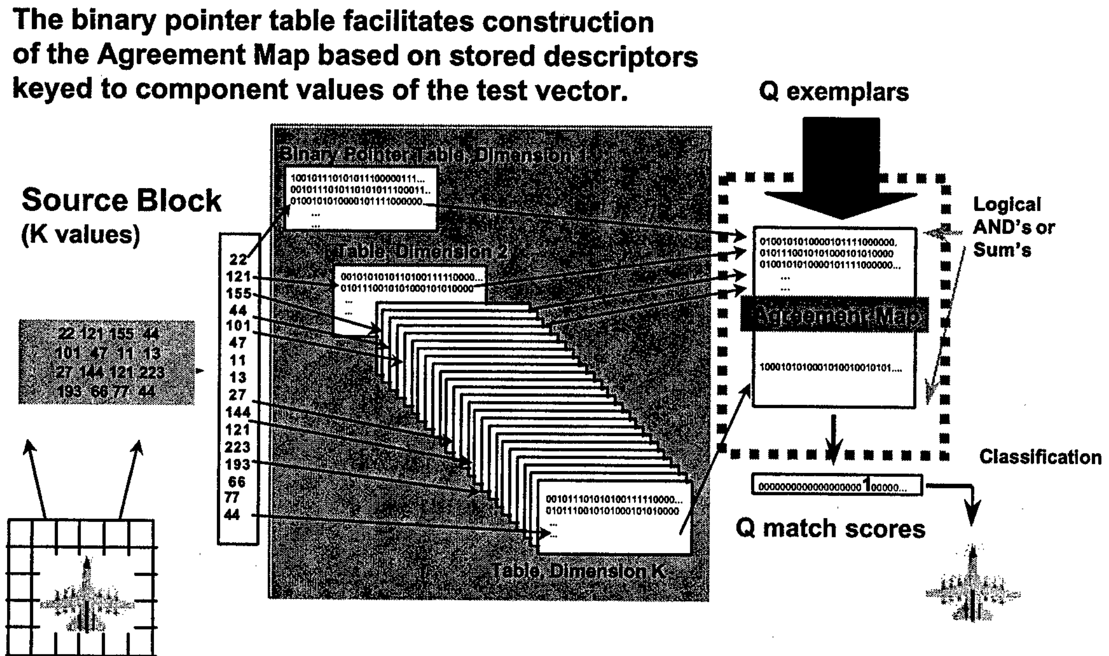


Figure 4.6: Schematic diagram of TNE as a target classifier.

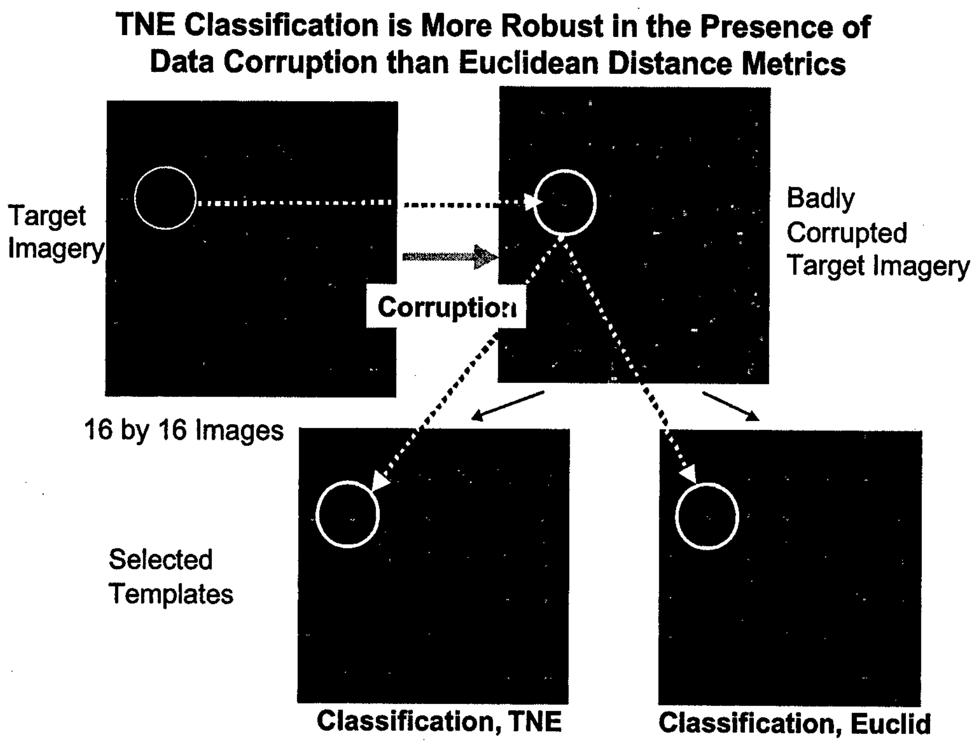

TNE Classification is More Robust in the Presence of Data Corruption than Euclidean Distance Metrics
Target Imagery
Badly Corrupted Target Imagery
Corruption
16 by 16 Images
Selected Templates
Classification, TNE
Classification, Euclid

Figure 4.7: Example of TNE-based classifier applied to noisy aircraft imagery.

It is possible to configure TNE hierarchically to produce more efficient search. For example, one could apply an operation O to a group of columns in an agreement map $\mathbf{m}_0$ to produce a reduced agreement map $\mathbf{m}_1$, then apply $\mathbf{m}_1$ to a source image **a** to produce approximate target classification. in categories $C_1 = \{c_{11}, c_{12}, \ldots, c_{1n}\}$. Without applying O to $\mathbf{m}_0$, one has categories $C_0 = \{ c_{01}, c_{02}, \ldots, c_{0m}\}$, where $c_{0i} \subset c_{1j}$ for some $1 \leq i \leq m$ and $1 \leq j \leq n$, with $n < m$. In preliminary tests, this clustering procedure produces accurate classification of targets in noisy imagery, similar to the results shown in Figure 4.7. We expect to analyze and present related test results in a future publication.

4.3.5. Example Application: TNE-Based Compression. Application of TNE encoding and search techniques to the training imagery shown in Figure 4.5 yielded an agreement map that was input to a TNE classification algorithm configured for image compression. The compression results for the *Lena* image are shown in Figure 4.8, and performance of the TNE algorithm, expressed as MSE versus CR, is graphed in Figure 4.9a. The latter figure instantiates the relationship between CR, MSE, and codebook size M, which is not apparent in the small sample shown in Figure 4.8.

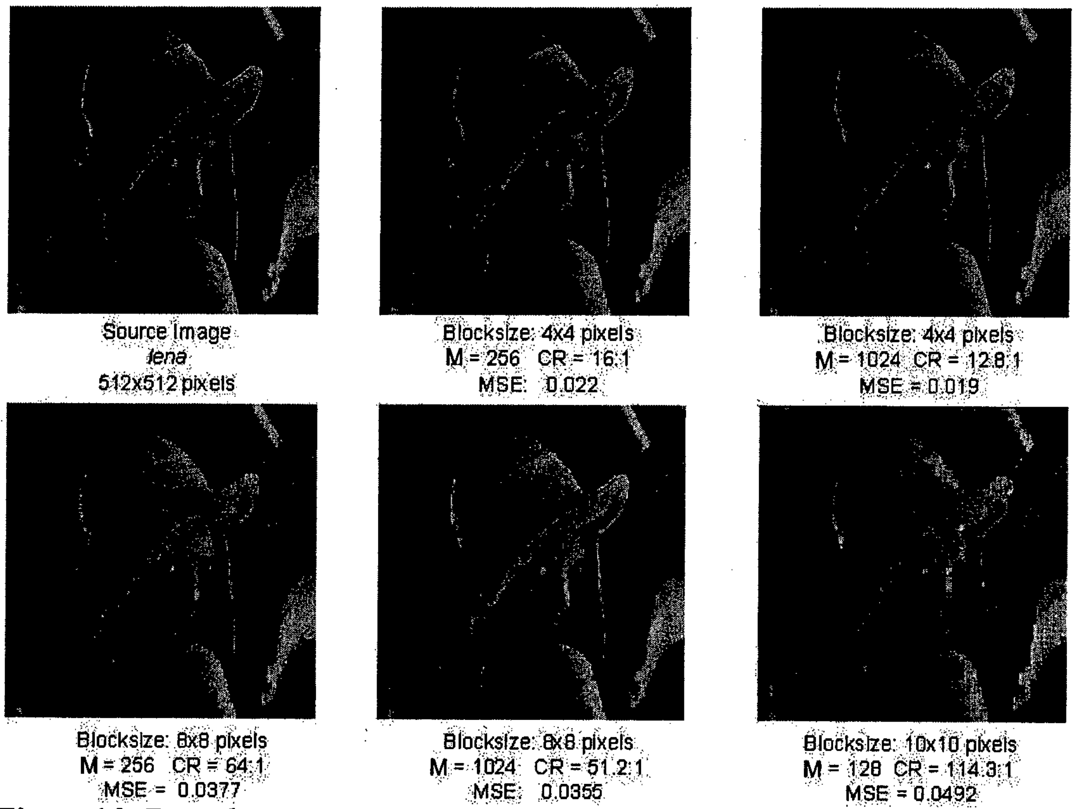
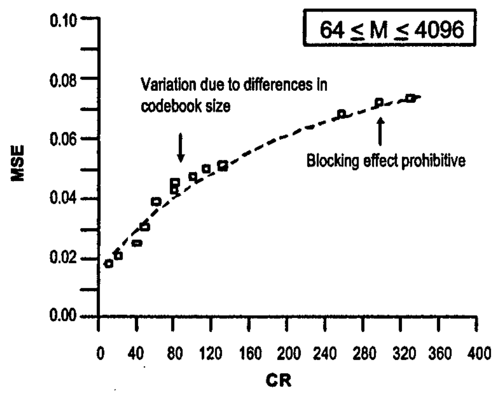


Figure 4.8: *Example TNE compression results on Lena image, where M denotes codebook size, mean-squared-error (MSE) is expressed as a fraction of full-scale intensity over the interval [0,1], and CR denotes compression ratio.*

In Figure 4.9a, MSE is shown to vary slightly more than **O**(CR), thus appearing to grow more slowly at higher compression ratios. This phenomenon, which is common to block-oriented compression transforms, is due to the rapid loss of high-frequency detail at low to moderate compression ratio (quantization in small to moderate blocks), followed by the gradual loss of low-frequency detail as blocksize increases. This is a physically valid observation, since low-frequency components predominate in natural imagery, where selected spatial information tends to scale as $1/f$, where f denotes spatial frequency [Wu94]. For example, compare the result obtained in Figure 4.8 with CR = 16:1 at blocksize 4x4 pixels with the result for 10x10-pixel blocks at CR = 114.3:1. The codebook sizes are within a factor of two; hence, the decompressed images are comparable. Note the loss of high-frequency detail and the induction of blocking effect in the latter image. As discussed previously, blocking effect is a severe problem in VQ-compressed imagery, which limits the achievable compression ratio to approximately 200:1 for 512x512 pixel imagery, which has 10x10-pixel blocks. The degradation of image quality as a result of blocking effect is noted in Figure 4.9a, and measures of such boundary effects are discussed in [Cai98], to which the reader is referred for further detail.



(a)

Method	M	RMSE	Time, sec
Euclidean	2048	0.022	258.8
TNE-1	2048	0.024	2.6
TNE-2	2048	0.023	4.9
TNE-3	2048	0.023	3.9
Euclidean	4096	0.021	441.5
TNE-1	4096	0.022	3.4
TNE-2	4096	0.022	4.5

(b)

Figure 4.9: *Example TNE compression performance on 512x512-pixel Lena image: (a) MSE versus CR for codebook size M ranging from 64 to 4096 exemplars, and (b) encoding times for Lena imagery of Figure 4.9 on a 133MHz Pentium running Microsoft Fortran.*

4.4 Heuristic Overview of TNE Paradigm for Data Fusion and ATR

We assume that the ATR process requires the comparison of multiple data blocks or *test vectors* from different sensor/classifier assets with a library of sought after target-related exemplar or *training set vectors.* We proceed by placing acceptance bounds around each component value of each test vector collected and determine the number of components from each training set exemplar that fall within these bounds. We generate a binary vector for each training set exemplar as was shown in Figure 4.3. This binary vector indicates the components of the test vector that agree with a particular database exemplar. Note that the intervals of agreement can be all equal or weighted as is appropriate. At the heart of the TNE paradigm is a procedure for very efficiently deriving this *agreement map* for the entire library without computation.

As showed in Figure 4.1, we construct a binary vector for each quantization level along each coordinate axis (vector block dimension). This binary vector contains a bit for each exemplar in the training set. If the N-space box corresponding to the agreement intervals about each training set exemplar vector point projects onto the particular quantization level, a 1 is assigned, otherwise a 0. The utility of these binary agreement vectors is that once they are constructed, a *virtual agreement map* for any new test vector can be specified immediately by simply using the N test vector component values as pointers into the larger, pre-derived table. Also, as shown in Figure 4.10, they can be combined logically to facilitate inferences

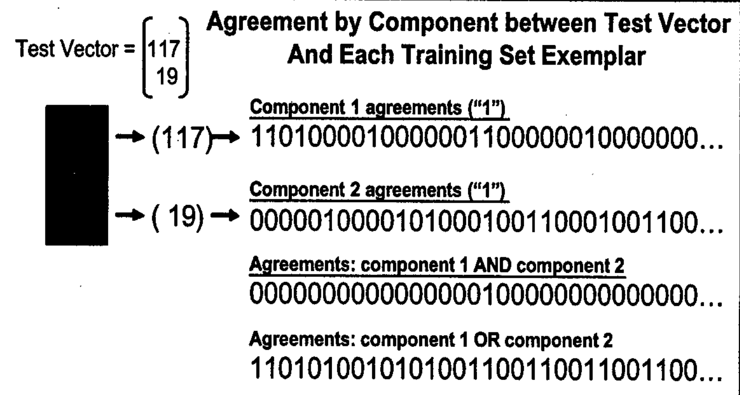

Test Vector = [117 19]
Agreement by Component between Test Vector And Each Training Set Exemplar
Component 1 agreements ("1")
(117) → 11010000100000011000000010000000...
Component 2 agreements ("1")
(19) → 00000100000101000100110001001100...
Agreements: component 1 AND component 2
00000000000000010000000000000000...
Agreements: component 1 OR component 2
11010100101010011001100110011100...

Figure 4.10 *Simple example: a two-dimensional system (i.e., two pixels per vector block) illustrates the utility of the binary agreement vectors. Note that only one Training Set exemplar matches both components of the test vector.*

with respect to test vector classification.

4.4.1 Data Fusion in TNE Paradigm

Note that the process outlined above is not limited to data from one sensor or sensor-type. We can readily accommodate multiple data sources by increasing the dimensionality of the signature vector to include components (e.g., samples, pixels, or state information) from all the available sources. When a particular source is not present, its representation in the search space is null.

For example, if three sensors collect K, M, and N samples, respectively, then the resulting query (test) vector will contain K+M+N components. We illustrate the specification of the

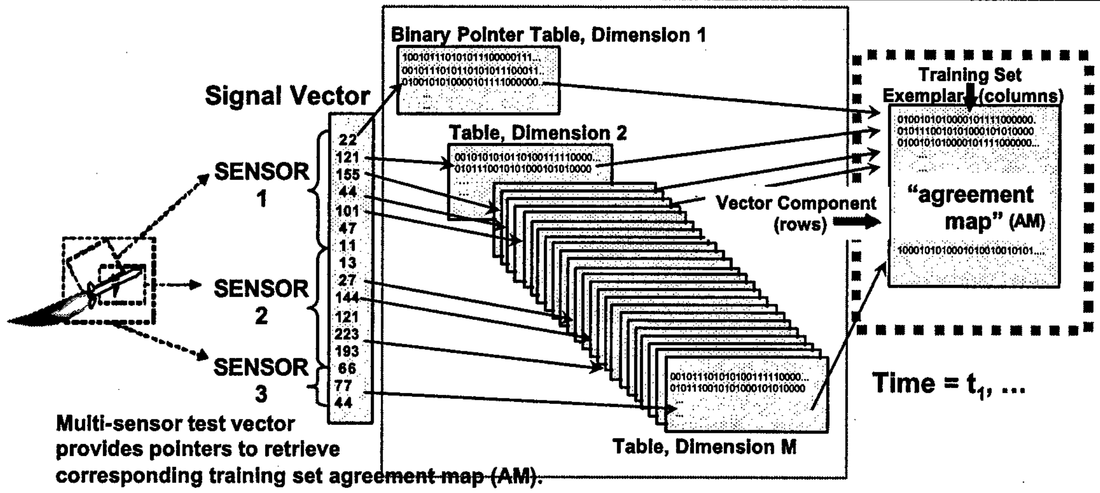

Binary Pointer Table, Dimension 1
Signal Vector
SENSOR 1
SENSOR 2
SENSOR 3
Table, Dimension 2
Training Set Exemplar (columns)
"agreement map" (AM)
Vector Component (rows)
Time = t1, ...
Table, Dimension M
Multi-sensor test vector provides pointers to retrieve corresponding training set agreement map (AM).

Figure 4.11a: *Specification of the single (virtual) Agreement Map via test vector component pointers, here derived from three fused sensor data types*

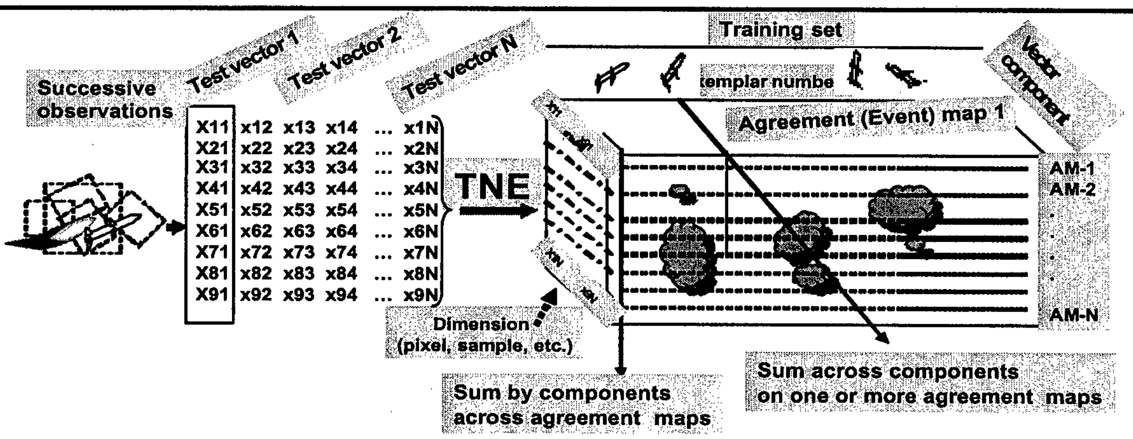


Figure 4.11b: *The TNE Algorithm associates input test vectors with their respective training set agreement maps (without any computation, only memory accesses). Multiple test vectors from sensor data give rise to sequences of agreement maps that together contain volume clusters of agreement events resulting from the data collection. These clusters support inferences with respect to target ID and/or false alarms.*

agreement map via the TNE algorithm in Figure 4.11a. Figure 4.11b illustrates how a sequence of observations (i.e., test vectors) leads directly (and at very low complexity) to an array or sequence of AMs associated with the specific collection sequence. We can think of this array (denoted by AM-1, AM-2, ...), where each AM corresponds to a test vector, as a higher-dimensional representation of agreement events indexed by training set exemplar and vector component, to be statistically analyzed as it evolves temporally. This 3-D binary structure records agreement events classified throughout a collection session. Target ID and false alarm inferences can subsequently be derived from observed clustering of the agreement events.

Figure 4.12 illustrates that much false alarm rejection can be achieved by dithering the AMs

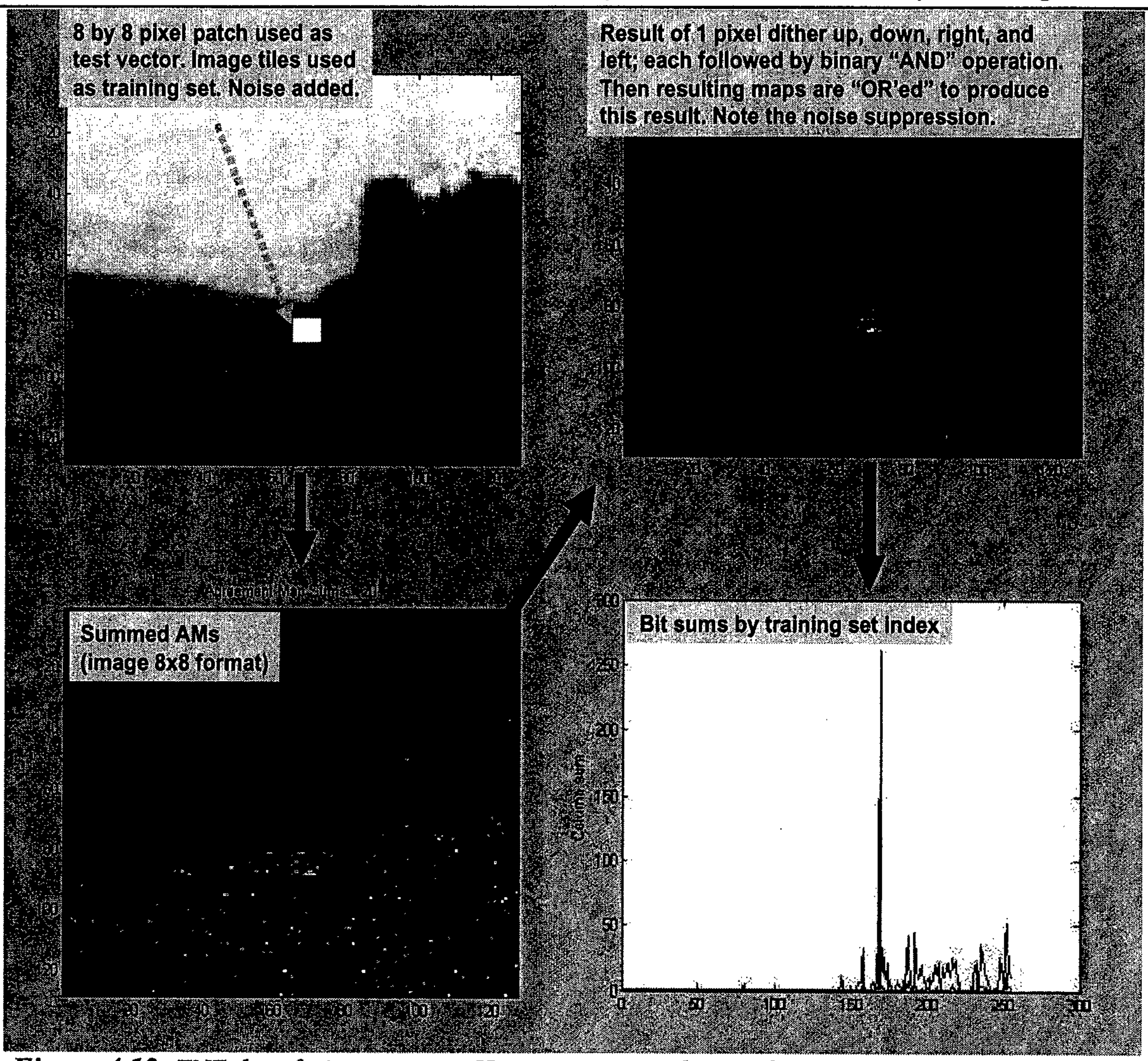

8 by 8 pixel patch used as test vector. Image tiles used as training set. Noise added.
Result of 1 pixel dither up, down, right, and left; each followed by binary "AND" operation. Then resulting maps are "OR'ed" to produce this result. Note the noise suppression.
Summed AMs (image 8x8 format)
Bit sums by training set index
0
50
100
150
200
250
300

Figure 4.12: *TNE data fusion process: Here, as an example, we tile an image into 8 by 8 pixel partitions, which for this example, we regard as a training set. We select a single image partition as a test vector and add noise to produce 20 realizations. We then use TNE to generate 20 agreement maps, which we sum and display as shown (lower left). We then dither each AM by one pixel in each rectangular direction and perform a binary AND operation on each pair. Finally, we apply the binary* or *operation on the four results, i.e., from dithering left, right, up, and down, as shown in the right-hand column. This procedure significantly reduces noise in the agreement event space. The resulting bit sum measure is shown at lower right. We emphasize that these results are obtained with little computational work.*

by one pixel and applying the binary *and* operation. We have found that this procedure exploits the important phenomenon of target spatial locality in ATR imagery. For example, we dither in each pixel direction (e.g., NEWS) and then apply the binary *or* operation to the results. The result is that at the cost of a moderate number of Boolean operations on a binary table, TNE thus yields a map of all non-isolated agreement events resulting from a collection sequence. Note too that most computer systems can do these Boolean operations 32 or 64 at a time (i.e., by applying the binary *bit* operations to two loaded 32- or 64-bit integers).

4.4.2 Data Fusion, AM Processing

Pure component voting of the agreement map entails summing each column to ascertain the number of agreements (i.e., N-Hamming distance) per reference pattern. As indicated, we sum the columns of the agreement map to obtain a voting score for each template in the training set. In some applications, it is advantageous to first apply the binary *or* operation to the columns related to closely correlated templates, for example, those corresponding to two configurations or articulations of the same target, prior to bit summing. In cases wherein multiple signal vectors are matched to clusters of templates from different candidate classes, we have found it useful to sort these sums while keeping track of their associated target IDs. We then select only the maximum few percentiles of these sums, i.e., templates that had a relatively good agreement score, and sum these by target ID or target class as desired.

4.4.3 Alternate Hardware Implementations

FTI teamed with University of Florida and Harbor Branch Oceanographic Institute on a contract effort (STTR N98T-003) with the Office of Naval Research (ONR) to implement TNE and other signal processing functions on reconfigurable computers (RCs), particularly field programmable gate arrays (FPGAs). Our objective in this effort was to leverage recent and on-going team foundational work to produce a surveillance processor with unprecedented speed, robustness, and adaptability. We show a simplified view of the processor architecture in Figure 4.13. As shown, each FPGA manages a portion of the agreement map. The components of the sensor vector(s) are passed to each FPGA block simultaneously where they are used as pointers into the stored binary table. In this example configuration, column sums are then done and the best result reported. Finally, the global best match is readout in real-time.

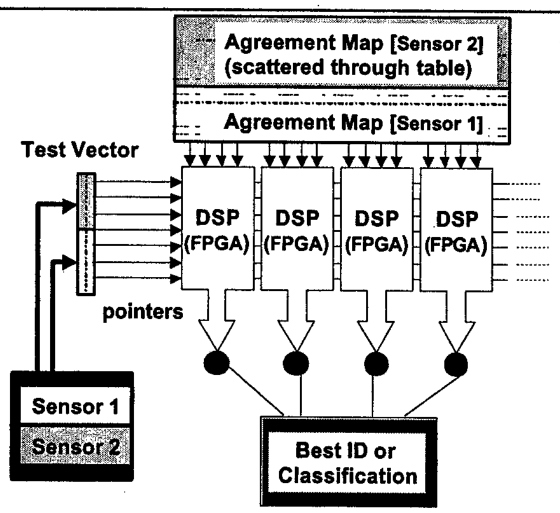


Figure 4.13: TNE implemented via a parallel array of FPGAs

4.4.4 TNE-Driven Inference Engine

To illustrate the TNE paradigm applied to a system of networked multiple ATR sensors and classifiers, consider the following configuration. We assume that each asset is continually collecting test signature vectors and sending these directly back to a TNE/MNN-based inference engine as network resources permit. At the central processing point, TNE is applied to each test vector from each asset in successive time slices. Depending on the signal to noise ratio, these collected vectors (data blocks) may compare poorly with any training set exemplars. This will result in random component matches in the resulting agreement maps, with an absence of identified targets. After the dither process described and illustrated above, the AMs would be expected to be virtually null. However, as the targets draw nearer to one or more recognizable assets (previously-mentioned effect of *spatial locality*), an increased incidence of agreements with associated training set exemplars is expected. Here, the contributions from different assets may be weighted prior to their combination, to account for changing environmental conditions, in order to increase Pd and decrease Rfa. The utility of the TNE AM structure in ATR inference is further discussed in Section 6.

4.5. References

[Cai98] Caimi, F.M., M.S. Schmalz, and G.X. Ritter. "Image quality measures for performance evaluation of compression transforms", *Proceedings SPIE* **3456**:136-152 (1998).

[Che99] Chen, W-S., F.C. Ou, L-C. Lin, and C. Hsin. "Image coding using vector quantization with a hierarchical codebook in wavelet domain", *IEEE Transactions on Consumer Electronics* **45**(1):36-45 (1999).

[ElS99] El-Sakka, M.R. and M.S. Kamel. "Adaptive image compression based on segmentation and block classification", *International Journal of Imaging Systems and Technology* **10**(1):33-46 (1999).

[Key99] Key, G., M.S. Schmalz, and F.M. Caimi. "Performance analysis of Tabular Nearest Neighbor Encoding for joint image compression and ATR", *Proceedings SPIE* **3814**:115-142 (1999).

[Mey98] Meyer, F.G., T.G. Averbuch, J.O. Stromberg, and R.R. Coifman. "Fast wavelet packet image compression", *Proceedings of the 1998 IEEE Data Compression Conference (DCC-98, Snowbird Utah)* p.563, (1998).

[Pog93] Poggi, G. "Fast algorithm for full-search VQ encoding", *Electronics Letters* **29**(12):1141-1142 (1993).

[Ram99] Ramasubramanian, V. and K.K. Paliwal. "Fast nearest-neighbor search based on Voronoi projections and its application to vector quantization encoding", *IEEE Transactions on Speech and Audio Processing* **7**(2):221-226 (1999).

[Sch97] Schmalz, M.S., G.X. Ritter, and F.M. Caimi. "Performance analysis of compression algorithms for noisy multispectral underwater imagery of small targets", *Proceedings SPIE* **3079**:191-102 (1997).

[Wu94] Wu, R-S. and Y-S. Lai. "Observation on the fractal characteristics of individual river plan forms", *IFIP Transactions A: Computer Science and Technology* **A-41**:441-451 (1994).

[Xu98] Xu, Y., Y. Liu, H. Chen, and D. Yisong. "Tree-structured vector quantization design using adaptive genetic algorithms", *Proceedings SPIE* **3561**:376-382 (1998).

5. MNN Pattern Recognition Paradigm

Morphological Neural Networks (MNNs) are an emerging pattern recognition technology that significantly improves upon the classification accuracy and performance characteristics of classical neural nets (CNNs). Morphological NNs are based on a nonlinear kernel that employs addition and maximum in a maximum-of-sums or minimum-of-sums configuration, versus classical NNs that employ multiplication and addition in a sum-of-products configuration. As a result of substituting addition for multiplication, the computational complexity is significantly reduced; MNNs can compute much faster than their classical counterparts. Also, classical NNs such as the Hopfield net have limited information storage capacity and generally unreliable pattern classification accuracy for all but a few highly distinct patterns. In contrast, associative memories based on MNNs can store and accurately retrieve the theoretical maximum number of patterns (2^N Boolean-valued patterns of length N) versus only 0.15N Boolean patterns for the Hopfield net [Lip87]. Because MNNs are based on an efficient kernel and converge in one pass of the net, they have short training times that support the use of MNNs in adaptive pattern recognition applications. Section 5.1 presents the mathematical theory of MNNs and morphological perceptrons, while Section 5.2 compares MNNs with previous and related research. Section 5.3 contains several applicative examples and complexity analysis. In Section 6, we show how TNE and MNNs can work together to improve pattern classification efficiency and accuracy in a Bayesian paradigm.

5.1. Mathematical Description of MNNs

We begin with an overview of CNN and MNN concepts, then progress to MNN perceptrons and learning algorithms. It is well known that content-addressable memories based on the Hopfield net have two key deficiencies. First, the number of patterns N that can be stored and accurately recalled is *severely* limited. If N is too large, then the net may converge to a spurious pattern that differs from all stored exemplars, which produces a "no-match" output in classifier applications [Lip87]. Hopfield remedied this defect by generating patterns *randomly* and keeping the number of classes $N_c < 0.15N$. For example, a Hopfield net for a simple ($N_c = 10$ class) problem would require more than 70 nodes and over 5,000 connection weights. Second, exemplar patterns in the Hopfield net are *unstable* if they share many bits in common with other exemplars. By unstable, we mean that after the exemplar is applied to the net's input, the net converges to a different exemplar.

5.1.1. Mathematical Overview.

The primary distinction between traditional or classical NNs and MNNs is the computation performed by an individual neuron. Computation represented by a classical NN (CNN) with a linear inner-product kernel (multiply-accumulate neuron) is based on the algebraic structure $(\mathbf{R},+,\cdot)$. The governing equation of a CNNs j^{th} neuron is given by

$$\mathrm{t}_j(\mathbf{x}) = \sum_{i=1}^{n} x_i w_{ij} - ?_j \,, \tag{5.1}$$

where the input vector is denoted by $\mathbf{x} \in \mathbf{R}^n$, x_i denotes the value of the i^{th} neuron, w_{ij} denotes the synaptic strength between the i^{th} and j^{th} neurons, θ_j denotes the threshold of the j^{th} neuron, and τ_j is the total input to the j^{th} neuron.

In contrast, morphological neural networks use lattice operations $\vee$ (maximum) and $\wedge$ (minimum), and + from the *semi-rings* $(\mathbf{R}_{-\infty},\vee,+)$ or $(\mathbf{R}_{\infty},\wedge,+')$. Here, the *extended real numbers* $\mathbf{R}_{-\infty} = \mathbf{R} \cup \{-\infty\}$ and $\mathbf{R}_{\infty} = \mathbf{R} \cup \{\infty\}$ have the following basic arithmetic and logic operations. The symbols + and +′ denote the usual addition operation with the stipulation that $a + (-\infty) = (-\infty) + a = -\infty \; \forall a \in \mathbf{R}_{-\infty}$. Similarly, $a +' \infty = \infty +' a = \infty \; \forall a \in \mathbf{R}_{\infty}$. The operations $\vee$ and $\wedge$ respectively carry the additional stipulation that $a \vee (-\infty) = (-\infty) \vee a = a \; \forall a \in \mathbf{R}_{-\infty}$ and $a \wedge \infty = \infty \wedge a = a \; \forall a \in \mathbf{R}_{\infty}$.

Note that the symbol $-\infty$ acts like a *zero* element in $(\mathbf{R}_{-\infty},\vee,+)$ if one views $\vee$ as *addition* and + as *multiplication.* Similar comments hold for ∞ in $(\mathbf{R}_{\infty},\wedge,+')$. Also, the role of the multiplicative identity in the structure $(\mathbf{R},+,\cdot)$ is filled by the number 1 (i.e., $1\cdot a = a\cdot 1 = a \;\; \forall a \in \mathbf{R}$). In the structures $(\mathbf{R}_{-\infty},\vee,+)$ and $(\mathbf{R}_{\infty},\wedge,+')$, this role is filled by 0, since $0 + a = a + 0 = a \;\; \forall a \in \mathbf{R}$.

Using $(\mathbf{R}_{-\infty},\vee,+)$, computation at a neuron in an MNN for input $\mathbf{x} = (x_1, \ldots, x_n)$ is given by

$$\mathrm{t}_j(\mathbf{x}) = p_j \bigvee_{i=1}^{n} r_{ij}\left(x_i + w_{ij}\right) \tag{5.2}$$

or

$$\mathrm{t}_j(\mathbf{x}) = p_j \bigwedge_{i=1}^{n} r_{ij}\left(x_i + w_{ij}\right) \tag{5.3}$$

where $r_{ij} = \pm 1$ denotes whether the i^{th} neuron causes excitation or inhibition on the j^{th} neuron, and $p_j = \pm 1$ denotes the output response (excitation or inhibition) of the j^{th} neuron. For excitatory input and output action, $r_{ij} = 1$ and $p_j = 1$, respectively. Inhibitory input and output responses have the values $r_{ij} = -1$ and $p_j = -1$, respectively. The computational model for a neuron in an MNN that uses the maximum operator is illustrated in Figure 5.1.

The activation function $f(\mathrm{t}_j)$ in an MNN is generally a hard limiter. Thus, the governing equation of a morphological neural net using the maximum operation is given by the following equation:

$$\mathrm{t}_j(\mathbf{x}) = f\left[p_j \bigvee_{i=1}^{n} r_{ij}\left(x_i + w_{ij}\right)\right].$$

Via the dual structure $(\mathbf{R}_{\infty},\wedge,+)$, the preceding equation becomes

$$\mathrm{t}_j(\mathbf{x}) = f\left[p_j \bigwedge_{i=1}^{n} r_{ij}\left(x_i + w_{ij}\right)\right].$$

The preceding two equations represent the basic morphological operations of *dilation* and *erosion* – hence the term *morphological* neural network.

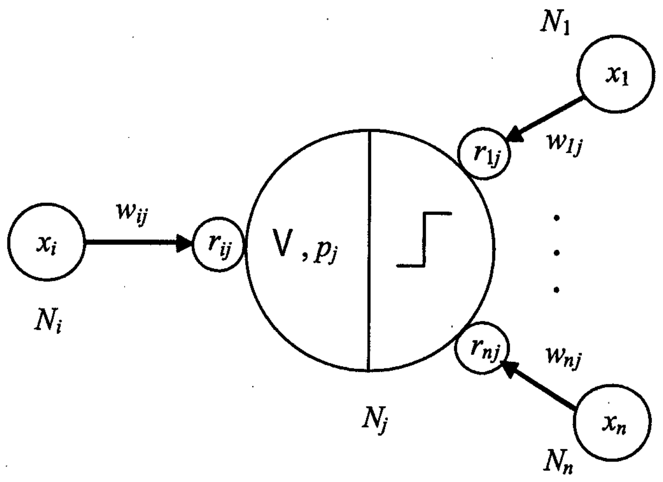


Figure 5.1: *Computational model of a morphological neuron.*

Several advantages accrue from replacing the arithmetic operations in traditional neural computing with lattice-based operations. It is apparent from Equations (5.2) and (5.3) that morphological neural computation does not involve multiplications but only the operations of logical *or* as well as *and* (respectively implemented as maximum and minimum over the Boolean numbers), together with addition and subtraction. This provides for extremely fast neural computation, and an algorithm that can be more readily implemented in hardware than a classical NN algorithm such as the Hopfield net. Convergence problems and lengthy training algorithms are often nonexistent [Rit97a,99a]. It has been shown that morphological associative memories are extremely robust in the presence of noise and have unlimited storage capacity [Rit97b,98,99b;RitXX,Sus00]. There is a close, natural connection between lattice based computing and fuzzy set theory, which makes lattice based networks amenable to handling more general data types and particular types of learning models [Pet98]. Finally, morphological neural networks are capable of solving any conventional computational problem and, thus, can be inherently useful in a wide variety of application domains [Rit96].

5.1.2. Lattice Algebra. In this discussion, we use the lattice algebra $(\mathbf{R}_\infty, \wedge, +)$ for our underlying neural computation. This algebra obeys the following laws: if $a,b,c \in \mathbf{R}_\infty$, then

$$\begin{aligned} a+(b\wedge c) &= (a+b)\wedge(a+c) \\ a\wedge\infty &= \infty\wedge a = a \\ a+\infty &= \infty+a=\infty \\ a+0 &= 0+a=a \end{aligned} \qquad (5.4)$$

To compare this system with the system $(\mathbf{R},+,\cdot)$ used in traditional neural computing, we have the analogous laws, if $a,b,c \in \mathbf{R}$, then

$$\begin{aligned} a\cdot(b+c) &= (a\cdot b)+(a\cdot c) \\ a+0 &= 0+a=a \\ a\cdot 0 &= 0\cdot a=0 \\ a\cdot 1 &= 1\cdot a=a \end{aligned} \qquad (5.5)$$

Thus, in the system $(\mathbf{R}_\infty, \wedge, +)$, the symbol ∞ acts as the null element and 0 as the unit element, if we view $\wedge$ as addition and + as multiplication.

In addition to the laws given in Equations (5.4) and (5.5), the two systems share many other properties such as commutativity and associativity. In this sense, the two systems are mathematically very similar and, therefore, one could be easily substituted for the other in many application areas. However, there are several differences, a major one is the lack of inverses for the lattice operation of minimum, namely $\wedge$. It is this lattice operation that provides for the unique and distinct properties of morphological neural networks.

It must also be mentioned that Koch and Poggio make a strong case for multiplying with synapses, i.e., for $x_i \cdot w_{ij}$ [Koc92].' In the model presented herein, multiplication could have been used just as well. A mathematical equivalent theory can be obtained by replacing the lattice $(\mathbf{R}_\infty, \wedge, +)$ with $(\mathbf{R}^{\geq 0}, \vee, +)$, where $\mathbf{R}^{\geq 0} = \{x \in \mathbf{R} : x \geq 0\}$. The mathematical equivalence is given by the isomorphism $\varphi : \mathbf{R}_\infty \to \mathbf{R}^{\geq 0}$ defined by $\varphi(x) = e^{-x}, \forall x \in \mathbf{R}$, and $\varphi(\infty) = 0$. It is readily seen that φ is a bijection. Furthermore, $\varphi(x+y) = \varphi(x) \cdot \varphi(y)$ and $\varphi(x \wedge y) = \varphi(x) \vee \varphi(y)$. Therefore, φ preserves the algebraic structure and is an isomorphism; note also that $\varphi(0) = 1$. Hence, identities are mapped to identities. The inverse of φ, denoted by $\varphi^{-1} : \mathbf{R}^{\geq 0} \to \mathbf{R}_\infty$, is given by $\varphi^{-1}(x) = -\ln(x)$ for $x > 0$ and $\varphi^{-1}(0) = \infty$. This proves that, mathematically, nothing is gained by replacing the additive lattice with the multiplicative lattice. Although a multiplicative lattice is more intuitive from a biological standpoint, a reason for using additive synapses is the reduction of computational cost and ease of describing the proposed (morphological) model of computation. This is especially true for emerging forms of morphological neural computing, for example, computing using single neurons with dendrites [Rit02].

Additionally, the additive lattice is directly related to mathematical morphology, a topic well known to the image processing and computer vision community [Dav92,Hei94]. We realize that some researchers are not familiar with the notion and uses of algebraically equivalent structures. In particular, questions may be raised pertaining to the meaning and use of negative weights, which arise quite naturally when using the lattice $(\mathbf{R}_\infty, \wedge, +)$. Traditionally, weight is a positive quantity with the sign determining whether the input is excitatory or inhibitory. Negative weights have nothing to do with inhibition in the additive lattice, where the sign of r_{ij} determines excitatory or inhibitory input. For a biological (or traditional) interpretation of negative weights, simply observe that these correspond to large positive weights in the equivalent multiplicative lattice $(\mathbf{R}^{\geq 0}, \wedge, \cdot)$ as $w = e^{-w_{ij}} > 0 \; \forall w_{ij} \in \mathbf{R}$ and $e^{-x} \to \infty$ as $x \to -\infty$.

The lattice $(\mathbf{R}_\infty, \wedge, +)$ has a natural dual (isomorphic) structure given by $(\mathbf{R}_{-\infty}, \vee, +)$, where $\forall x \in \mathbf{R}_{-\infty}$, $x + (-\infty) = (-\infty) + x = -\infty$. For each $\forall x \in \mathbf{R}_{\pm\infty} = \mathbf{R} \cup \{-\infty, \infty\}$, we define its dual or conjugate by $x^* = -x$, where $-(-\infty) = \infty$. The following duality laws are a direct consequence of this definition:

$$\begin{aligned} (x^*)^* &= x \\ (x \wedge y)^* &= x^* \vee y^* \\ (x \vee y)^* &= x^* \wedge y^* \end{aligned} \tag{5.6}$$

The function $\psi : \mathbf{R}_{-\infty} \to \mathbf{R}_{\infty}$ defined by $\psi(x) = x^*$ is an isomorphism. Therefore, a morphological neural net using the operation $\vee$ can always be reformulated in terms of the operation $\wedge$, and vice versa. Due to the relations $-(x \wedge y) = -x \vee -y$ and $-x \wedge -y = -(x \vee y)$ of Equation (5.6), and the use of inhibitory input and output, the algebra $(\mathbf{R}_{-\infty}, \vee, +)$ is implicitly incorporated in Dr. Ritter's current research in dendritic computation in morphological neurons [Rit02].

5.1.3. Morphological Operations. Given the computations represented by Equations (5.2) and (5.3), let $\mathbf{v}'$ denote the transpose of the vector $\mathbf{v}$. The total network computation for a matrix memory such as a Hopfield or Kohonen memory with input $\mathbf{a}$ at time t can be expressed in matrix form as

$$T(t+1) = W \cdot \mathbf{a}(t)\,, \tag{5.7}$$

where $\mathbf{a}(t) = (a_1(t), \cdots, a_n(t))'$, $T(t+1) = (\tau_1(t+1), \cdots, \tau_n(t+1))'$, and W denotes the nxn synaptic weight matrix whose i,jth entry is w_{ij}. Analogous to Equation (5.7), the total morphological network computation for a morphological matrix memory can also be expressed in matrix form. In order to do this, we need to define a matrix product in terms of the operations of the lattice structure $(\mathbf{R}, \vee, \wedge, +)$. For an *m*x*p* matrix *A* and a *p*x*n* matrix *B* with entries from **R**, the matrix product $C = A \boxed{\vee} B$, also called the *max product* of *A* and *B*, is defined by

$$c_{ij} = \bigvee_{k=1}^{p} a_{ik} + b_{kj} = \left(a_{i1} + b_{1j}\right) \vee \left(a_{i2} + b_{2j}\right) \vee \cdots \vee \left(a_{ip} + b_{pj}\right). \tag{5.8}$$

The *min product* of *A* and *B* induced by the lattice structure is defined in a similar fashion. Specifically, the *i,j*th entry of $C = A \boxed{\wedge} B$ is given by

$$c_{ij} = \bigwedge_{k=1}^{p} a_{ik} + b_{kj} = \left(a_{i1} + b_{1j}\right) \wedge \left(a_{i2} + b_{2j}\right) \wedge \cdots \wedge \left(a_{ip} + b_{pj}\right). \tag{5.9}$$

The total MNN computation for an associative memory can now be expressed in the matrix forms

$$T(t+1) = W \boxed{\vee} \mathbf{a}(t) \tag{5.10}$$

and

$$T(t+1) = W \boxed{\wedge} \mathbf{a}(t) \tag{5.11}$$

respectively.

Some additional comments concerning lattice-based operations are pertinent when discussing morphological network computations. When using the lattice $(\mathbf{R}, \vee, \wedge +)$, the maximum (or minimum) of two matrices replaces the usual matrix addition of linear algebra. Here, the *i,j*th entry of the matrix $C = A \vee B$ is given by $c_{ij} = a_{ij} \vee b_{ij}$. Similarly, the minimum of two matrices $C = A \wedge B$ is defined by $c_{ij} = a_{ij} \wedge b_{ij}$. Finally, we say that *A is less or equal than B,* denoted by $A \leq B$, and *A is strictly less than B*, denoted by $A < B$, if and only if for each corresponding entries of these matrices we have that $a_{ij} \leq b_{ij}$ and $a_{ij} < b_{ij}$, respectively.

The algebraic structure $(\mathbf{R}, \vee, \wedge +)$ provides for an elegant duality between matrix operations. For any real number *r*, we define its *additive conjugate* $r^* = -r$. Therefore,

$$(r^*)^* = r \quad \text{and} \quad r \wedge u = (r^* \vee u^*)^*, \;\; \forall r, u \in \mathbf{R}\,. \tag{5.12}$$

Now, if $A = (a_{ij})_{mxn}$ is an m x n matrix with $a_{ij} \in \mathbf{R}$, then the *conjugate matrix* A^* *of* A is the n x m matrix $A^* = (b_{ij})_{mxn}$ defined by $b_{ij} = [a_{ji}]^*$, where $[a_{ji}]^*$ denotes the additive conjugate of a_{ji}. It follows that

$$A \wedge B = (A^* \vee B^*)^* \tag{5.13}$$

and

$$A \boxed{\wedge} B = (A^* \boxed{\vee} B^*)^* \tag{5.14}$$

for appropriately sized matrices. This implies that a morphological neural net using the operation $\boxed{\vee}$ [i.e., Equation (5.10)], can always be reformulated in terms of the operation $\boxed{\wedge}$ [i.e., Equation (5.11)], and vice versa, by using the duality relations expressed in Equations (5.13) and (5.14).

Having defined the necessary mathematical tools, we next discuss basick properties of morphological neural networks and present some examples.

5.1.4. Morphological Associative Memories. The ability of human beings to retrieve information on the basis of associated cues continues to elicit great interest among researchers. For example, a few pictures from a movie clip can evoke memory of the entire movie's plot. Similarly, a glimpse of a partially occluded face in a crowd can be sufficient basis for recognizing an old friend. Investigations of how the brain is capable of constructing such associations from partial information have led to a variety of theoretical neural-network models that act as associative memories. The basic goal of these artificial associative memories is the retrieval of complete sorted pattern from noisy or incomplete input pattern keys. These associative memories are categorized as hetereoassociative or autoassociative, with the latter comprising the primary focus of this discussion. Among all autoassociative networks the Hopfield network is the most widely known [Hop82a-b,86]. A large number of researchers have exhaustively studied this network, its variations, and generalizations [Abu85,Ami85,Che86, Den86,Kee86,Kos87a-b,McE87].

In classical neural network-theory, for a given input vector or *key* $\mathbf{x} = (x_1, \cdots, x_n)'$, an *associative memory* W recalls a vector output signal $f(\mathbf{y})$, where $\mathbf{y} = W \cdot \mathbf{x}$. If $f(y_i) = y_i \; \forall i$, then $f(\mathbf{y}) = \mathbf{y}$ and the memory is called a *linear associative memory*, otherwise it is referred to as a *semilinear associative memory*. A basic question concerning associative memories is: What is the simplest way to store k vector pairs $(\mathbf{x}^1, \mathbf{y}^1), \cdots, (\mathbf{x}^k, \mathbf{y}^k)$, where $\mathbf{x}^\xi \in \mathbf{R}^n$ and $\mathbf{y}^\xi \in \mathbf{R}^m$ for $\xi = 1, \cdots, k$ in an m x n memory W? The well-known answer to thtat question for linear and semilinear associative memories is to set

$$W = \sum_{\xi=1}^{k} \mathbf{y}^\xi \cdot (\mathbf{x}^\xi)'. \tag{5.15}$$

In this case, the i,jth entry of W is given by $w_{ij} = \sum_{\xi=1}^{k} y_i^\xi x_j^\xi$. If the input patterns $\mathbf{x}^1, \cdots, \mathbf{x}^k$ are orthonormal, that is

$$(x^j)' \cdot x^i = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}, \tag{5.16}$$

then

$$W \cdot \mathbf{x}^i = \left(\mathbf{y}^1 \cdot (\mathbf{x}^1)' + \cdots + \mathbf{y}^k \cdot (\mathbf{x}^k)'\right) \cdot \mathbf{x}^i = \mathbf{y}^i. \tag{5.17}$$

Thus, we have *perfect recall* of the output patterns $\mathbf{y}^1,\cdots,\mathbf{y}^k$. If $\mathbf{x}^1,\cdots,\mathbf{x}^k$ are not orthonormal (which they are not in most realistic cases), then the *noise term*

$$N = \sum_{i \neq j} \mathbf{y}^j \cdot ((\mathbf{x}^j)' \cdot \mathbf{x}^i) \neq 0 \tag{5.18}$$

contributes to *crosstalk* in the recalled pattern by additively modulating the signal term. Therefore, filtering processes using threshold functions become necessary in order to retrieve the desired output pattern.

Morphological associative memories, which are based on the lattice algebra described in Sections 5.2 and 5.3, are surprisingly similar to these classical associative memories. Suppose we are given a vector pair $\mathbf{x} = (x_1,\cdots,x_n)' \in \mathbf{R}^n$ and $\mathbf{y} = (y_1,\cdots,y_m)' \in \mathbf{R}^m$. An associative morphological memory that will recall the vector $\mathbf{y}$ when presented the vector $\mathbf{x}$ is given by

$$W = \mathbf{y} \boxed{\vee} (-\mathbf{x})' = \begin{pmatrix} y_1 - x_1 & \cdots & y_1 - x_n \\ \vdots & \ddots & \vdots \\ y_m - x_1 & \cdots & y_m - x_n \end{pmatrix} \tag{5.19}$$

since W satisfies the equation $W \boxed{\vee} \mathbf{x} = \mathbf{y}$, as can be verified by the simple computation

$$W \boxed{\vee} \mathbf{x} = \begin{pmatrix} \bigvee_{i=1}^{n} (y_1 - x_i + x_i) \\ \vdots \\ \bigvee_{i=1}^{n} (y_m - x_i + x_i) \end{pmatrix} = \mathbf{y}. \tag{5.20}$$

Note the similarity between Equations (5.15) and (5.19) when $k = 1$. The natural question one may ask is whether or not this concept can be extended to cases where $k > 1$. The answer is a qualified "yes", due to the fact that problems which in some way are analogous to those associated with ordinary associative memories [i.e., Equation (5.18)] also occur in morphological associative memories.

Henceforth, let $(\mathbf{x}^1,\mathbf{y}^1),\cdots,(\mathbf{x}^k,\mathbf{y}^k)$ be k vector pairs with where $\mathbf{x}^\xi = (x_1^\xi,\cdots,x_n^\xi) \in \mathbf{R}^n$ and $\mathbf{y}^\xi = (y_1^\xi,\cdots,y_n^\xi) \in \mathbf{R}^m$ for $\xi = 1,\cdots,k$. For a given set of pattern associations $\{(\mathbf{x}^\xi,\mathbf{y}^\xi) : \xi = 1,\cdots,k\}$ we define a pair of associated pattern matrices (X,Y), where $X = (\mathbf{x}^1,\cdots,\mathbf{x}^k)$ and $Y = (\mathbf{y}^1,\cdots,\mathbf{y}^k)$. Thus, X is of dimension n x k with *i,j*th entry x_i^j and Y is of dimension m x k with *i,j*th entry y_i^j.

Since $\mathbf{y}^\xi \boxed{\vee} (-\mathbf{x}^\xi)' = \mathbf{y}^\xi \boxed{\wedge} (-\mathbf{x}^\xi)'$, we reduce notational burden by denoting these identical morphological outer vector products by $\mathbf{y}^\xi \times (-\mathbf{x}^\xi)'$. With each pair of matrices (X,Y), we associate two natural morphological m x n memories W_{XY} and M_{XY}, defined by

$$W_{XY} = \bigwedge_{\xi=1}^{k} \left[\mathbf{y}^\xi \times (-\mathbf{x}^\xi)'\right] \quad \text{and} \quad M_{XY} = \bigvee_{\xi=1}^{k} \left[\mathbf{y}^\xi \times (-\mathbf{x}^\xi)'\right]. \tag{5.21}$$

It follows from this definition that

$$W_{XY} \leq \mathbf{y}^\xi \times (-\mathbf{x}^\xi)' \leq M_{XY} \quad \forall \xi = 1,\cdots,k. \tag{5.22}$$

In view of Equation (5.20), this last set of inequalities implies that

$$W_{XY} \boxed{\vee} \mathbf{x}^{\xi} \le \left[\mathbf{y}^{\xi} \times (-\mathbf{x}^{\xi})'\right] \boxed{\vee} \mathbf{x}^{\xi} = \mathbf{y}^{\xi} = \left[\mathbf{y}^{\xi} \times (-\mathbf{x}^{\xi})'\right] \le M_{XY} \boxed{\wedge} \mathbf{x}^{\xi} \tag{5.23}$$

$\forall \xi = 1, \cdots . k$ or, equivalently, that

$$W_{XY} \boxed{\vee} X \le Y \le M_{XY} \boxed{\wedge} X \,. \tag{5.24}$$

Example 1. Let

$$\mathbf{x}^1 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{y}^1 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \quad \mathbf{x}^2 = \begin{pmatrix} 0 \\ -2 \\ -4 \end{pmatrix}, \quad \mathbf{y}^2 = \begin{pmatrix} -1 \\ -1 \\ 0 \end{pmatrix}, \quad \mathbf{x}^3 = \begin{pmatrix} 0 \\ -3 \\ 0 \end{pmatrix}, \quad \mathbf{y}^3 = \begin{pmatrix} 0 \\ -2 \\ 0 \end{pmatrix}. \tag{5.25}$$

According to Equation (5.21), the memories W_{XY} *and* M_{XY} are given by

$$\begin{aligned} W_{XY} &= \bigwedge_{\xi=1}^{3} \left[\mathbf{y}^{\xi} \times (-\mathbf{x}^{\xi})'\right] \\ &= \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix} \wedge \begin{pmatrix} -1 & 1 & 3 \\ -1 & 1 & 3 \\ 0 & 2 & 4 \end{pmatrix} \wedge \begin{pmatrix} 0 & 3 & 0 \\ -2 & 1 & -2 \\ 0 & 3 & 0 \end{pmatrix} \\ &= \begin{pmatrix} -1 & 0 & 0 \\ -2 & 1 & -2 \\ 0 & 0 & 0 \end{pmatrix} \end{aligned} \tag{5.28}$$

and

$$\begin{aligned} M_{XY} &= \bigvee_{\xi=1}^{3} \left[\mathbf{y}^{\xi} \times (-\mathbf{x}^{\xi})'\right] \\ &= \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix} \vee \begin{pmatrix} -1 & 1 & 3 \\ -1 & 1 & 3 \\ 0 & 2 & 4 \end{pmatrix} \vee \begin{pmatrix} 0 & 3 & 0 \\ -2 & 1 & -2 \\ 0 & 3 & 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 & 3 & 3 \\ 1 & 1 & 3 \\ 0 & 3 & 4 \end{pmatrix} \end{aligned} \tag{5.29}$$

It can be easily verified that $W_{XY} \boxed{\vee} \mathbf{x}^{\xi} = \mathbf{y}^{\xi} = M_{XY} \boxed{\wedge} \mathbf{x}^{\xi}$ holds for ξ = 1,2,3. For example,

$$W_{XY} \boxed{\vee} \mathbf{x}^1 = \begin{pmatrix} -1 & 0 & 0 \\ -2 & 1 & -2 \\ 0 & 0 & 0 \end{pmatrix} \boxed{\vee} \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \mathbf{y}^1 \tag{5.30}$$

and

$$M_{XY} \boxed{\wedge} \mathbf{x}^2 = \begin{pmatrix} 0 & 3 & 3 \\ 1 & 1 & 3 \\ 0 & 3 & 4 \end{pmatrix} \boxed{\wedge} \begin{pmatrix} 0 \\ -2 \\ -4 \end{pmatrix} = \begin{pmatrix} -1 \\ -1 \\ 0 \end{pmatrix} = \mathbf{y}^2 \,. \tag{5.31}$$

Definition 1. A matrix $A=(a_{ij})_{m\times n}$ is said to be a $\boxed{\vee}$*-perfect* memory for (X,Y) if and only if $A\boxed{\vee}X=Y$. The matrix A is said to be a $\boxed{\wedge}$*-perfect* memory for (X,Y) if and only if $A\boxed{\wedge}X=Y$. Obviously, A is $\boxed{\vee}$-perfect for (X,Y) if and only if $A\boxed{\vee}\mathbf{x}^{\xi}=\mathbf{y}^{\xi}$ for $\xi=1,\cdots,k$. Similarly, A is $\boxed{\wedge}$-perfect for (X,Y) if and only if $A\boxed{\wedge}\mathbf{x}^{\xi}=\mathbf{y}^{\xi}$ for $\xi=1,\cdots,k$. Also, if A is $\boxed{\vee}$-perfect for (X,Y), then $\left(A\boxed{\vee}\mathbf{x}^{\xi}\right)_i=\mathbf{y}_i^{\xi}$ for all $\xi=1,\cdots,k$ and all $i=1,\cdots,m$. Equivalently,

$$\bigvee_{j=1}^{n}\left(a_{ij}+x_j^{\xi}\right)=y_i^{\xi}\ \forall\xi=1,\cdots,k \text{ and } i=1,\cdots,m. \tag{5.32}$$

It follows that, for an arbitrary index $j\in\{1,\cdots,n\}$, we have

$$\begin{aligned} & a_{ij}+x_j^{\xi}\le y_i^{\xi}\ \ \forall\xi=1,\cdots,k \\ & \Leftrightarrow a_{ij}\le y_i^{\xi}-x_j^{\xi}\ \ \forall\xi=1,\cdots,k \\ & \Leftrightarrow a_{ij}\le\bigwedge_{\xi=1}^{k}\left(y_i^{\xi}-x_j^{\xi}\right)=w_{ij} \end{aligned} \tag{5.33}$$

This shows that $A \le W_{XY}$. In view of Equation (5.26), we now have $Y=A\boxed{\vee}X\le W_{XY}\boxed{\vee}X\le Y$ and, therefore, $W_{XY}\boxed{\vee}X=Y$. A similar argument shows that if B is $\boxed{\wedge}$-perfect for (X,Y), then $M_{XY}\le B$ and $M_{XY}\boxed{\wedge}X=Y$. This establishes the following optimality criteria for morphological associative memories:

Theorem 1. *If A is $\boxed{\vee}$-perfect for (X,Y) and B is $\boxed{\wedge}$-perfect for (X,Y), then*

$$A\le W_{XY}\le M_{XY}\le B \quad\text{and}\quad W_{XY}\boxed{\vee}X=Y=M_{XY}\boxed{\wedge}X. \tag{5.34}$$

According to this theorem, W_{XY} is the least upper bound of all $\boxed{\vee}$-perfect recall memories and M_{XY} is the greatest lower bound of all $\boxed{\wedge}$-perfect recall memories for (X,Y). Furthermore, if there exists a $\boxed{\vee}$-perfect ($\boxed{\wedge}$-perfect) recall memory, then W_{XY} (resp. M_{XY}) is also a perfect recall memory.

The next obvious question concerns the existence of a perfect recall memory. Specifically, for what vector pairs $(\mathbf{x}^1,\mathbf{y}^1)$, …, $(\mathbf{x}^k,\mathbf{y}^k)$ will W_{XY} or M_{XY} provide perfect recall? Once this question has been answered, the next logical question pertains to the amount of noise W_{XY} or M_{XY} can tolerate for perfect recall; i.e., if $\tilde{\mathbf{x}}^{\xi}$ denotes a distorted version of $\mathbf{x}^{\xi}$, what are the conditions or bounds on $\tilde{\mathbf{x}}^{\xi}$ to ensure that $W_{XY}\boxed{\vee}\tilde{\mathbf{x}}^{\xi}=\mathbf{y}^{\xi}$ or $M_{XY}\boxed{\wedge}\tilde{\mathbf{x}}^{\xi}=\mathbf{y}^{\xi}$? We provide answers to these questions through a series of theorems and corollaries. Proof of these theorems are given in [Rit98].

The following theorem answers the existence question of perfect recall memories for sets of pattern pairs.

Theorem 2. *W_{XY} is $\boxed{\vee}$-perfect for (X,Y) if and only if, for each $\xi=1,\cdots,k$, each row of the matrix $[\mathbf{y}^{\xi}\times(-\mathbf{x}^{\xi})']-W_{XY}$ contains a zero entry. Similarly, M_{XY} is $\boxed{\wedge}$-perfect for (X,Y) if and only if, for each $\xi=1,\cdots,k$, each row of the matrix $M_{XY}-[\mathbf{y}^{\xi}\times(-\mathbf{x}^{\xi})']$ contains a zero entry.*

Accordingly, $W_{XY} \boxtimes \mathbf{x}^{\xi} = \mathbf{y}^{\xi}\ \forall \xi = 1,\cdots,k$ if and only if for each ξ and each row index i there exists a column index j (depending on ξ and i) such that $w_{ij} = [\mathbf{y}^{\xi} \times (-\mathbf{x}^{\xi})']_{ij}$, where w_{ij} denotes the i,jth entry of W_{XY}. Similarly, $M_{XY} \boxed{\wedge}\ \mathbf{x}^{\xi} = \mathbf{y}^{\xi}\ \forall \xi = 1,\cdots,k$ if and only if for each ξ and each row index i there exists a column index j (depending on ξ and i) such that $w_{ij} = [\mathbf{y}^{\xi} \times (-\mathbf{x}^{\xi})']_{ij}$.

Example 2. The memories W_{XY} and M_{XY} given in Example 1 satisfy the conditions of Theorem 2. For example, for $\xi = 2$, we have

$$\begin{aligned} w_{11} &= y_1^2 - x_1^2 = [\mathbf{y}^2 \times (-\mathbf{x}^2)']_{11} \\ w_{22} &= y_2^2 - x_2^2 = [\mathbf{y}^2 \times (-\mathbf{x}^2)']_{22} \\ w_{31} &= y_3^2 - x_1^2 = [\mathbf{y}^2 \times (-\mathbf{x}^2)']_{31} \end{aligned} \tag{5.35}$$

and

$$\begin{aligned} m_{13} &= y_1^2 - x_3^2 = [\mathbf{y}^2 \times (-\mathbf{x}^2)']_{13} \\ m_{22} &= y_2^2 - x_2^2 = [\mathbf{y}^2 \times (-\mathbf{x}^2)']_{22} \, . \\ m_{33} &= y_3^2 - x_3^2 = [\mathbf{y}^2 \times (-\mathbf{x}^2)']_{33} \end{aligned} \tag{5.36}$$

The following is an easy consequence of Theorem 2.

Corollary 2.1. $W_{XY} \boxtimes X = Y$ if and only if for each row index $i = 1,\cdots,m$ and each $\gamma \in \{1,\cdots,k\}$ there exists a column index $j \in \{1,\cdots,n\}$ depending on i and γ such that

$$x_j^{\gamma} = \bigvee_{\xi=1}^{k} (x_j^{\xi} - y_i^{\xi}) + y_i^{\gamma} . \tag{5.37}$$

$M_{XY} \boxed{\wedge} X = Y$ if and only if for each row index $i = 1,\cdots,m$ and each $\gamma \in \{1,\cdots,k\}$ there exists a column index $j \in \{1,\cdots,n\}$ depending on i and γ such that

$$x_j^{\gamma} = \bigwedge_{\xi=1}^{k} (x_j^{\xi} - y_i^{\xi}) + y_i^{\gamma} . \tag{5.38}$$

The next theorem provides bounds for the amount of distortion of the exemplar patterns $\mathbf{x}^{\xi}$ for which perfect recall can be assured.

Theorem 3. Let $\tilde{\mathbf{x}}^{\gamma}$ denote a distorted version of the pattern $\mathbf{x}^{\gamma}$. Then $W_{XY} \boxtimes \tilde{\mathbf{x}}^{\gamma} = \mathbf{y}^{\gamma}$ if and only if

$$\tilde{x}_j^{\gamma} \le x_j^{\gamma} \vee \bigwedge_{i=1}^{m} \left(\bigvee_{\xi \ne \gamma} [y_i^{\gamma} - y_i^{\xi} + x_j^{\xi}] \right) \quad \forall j = 1,\cdots,n \tag{5.39}$$

and for each row index $i \in \{1,\cdots,m\}$ there exists a column index $j_i \in \{1,\cdots,n\}$ such that

$$\tilde{x}^{\gamma}_{j_i} = x^{\gamma}_{j_i} \vee \left(\bigvee_{\xi \neq \gamma} [y^{\gamma}_i - y^{\xi}_i + x^{\xi}_{j_i}] \right). \tag{5.40}$$

Similarly, $M_{XY} \boxtimes \tilde{\mathbf{x}}^{\gamma} = \mathbf{y}^{\gamma}$ if and only if

$$\tilde{x}^{\gamma}_j \geq x^{\gamma}_j \wedge \bigvee_{i=1}^{m} \left(\bigwedge_{\xi \neq \gamma} [y^{\gamma}_i - y^{\xi}_i + x^{\xi}_j] \right) \quad \forall j = 1, \cdots, n \tag{5.41}$$

and for each row index $i \in \{1, \cdots, m\}$ there exists a column index $j_i \in \{1, \cdots, n\}$ such that

$$\tilde{x}^{\gamma}_{j_i} = x^{\gamma}_{j_i} \wedge \left(\bigwedge_{\xi \neq \gamma} [y^{\gamma}_i - y^{\xi}_i + x^{\xi}_{j_i}] \right). \tag{5.42}$$

The subscript notation j_i denotes the dependence of the column index j on the particular row index i.

Example 3. Consider the following set of associated vector pairs:

$$\mathbf{x}^1 = \begin{pmatrix} -5 \\ -6 \\ -1 \end{pmatrix}, \quad \mathbf{y}^1 = \begin{pmatrix} -1 \\ 1 \\ -3 \end{pmatrix}, \quad \mathbf{x}^2 = \begin{pmatrix} 0 \\ -4 \\ -1 \end{pmatrix}, \quad \mathbf{y}^2 = \begin{pmatrix} 2 \\ 5 \\ -1 \end{pmatrix}, \quad \mathbf{x}^3 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{y}^3 = \begin{pmatrix} 5 \\ 7 \\ 0 \end{pmatrix}. \tag{5.43}$$

If $\tilde{\mathbf{x}}^1 = (-4, -8, -1)'$, then $\tilde{x}^1_j \leq x^1_j \vee \bigwedge_{i=1}^{3} \left(\bigvee_{\xi=2}^{3} [y^1_i - y^{\xi}_i + x^{\xi}_j] \right)$ for j = 1, 2, and 3. For instance,

$$\begin{aligned} & x^1_1 \vee \bigwedge_{i=1}^{3} \left(\bigvee_{\xi=2}^{3} [y^1_i - y^{\xi}_i + x^{\xi}_1] \right) \\ & = x^1_1 \vee \bigwedge_{i=1}^{3} \left[\left(y^1_i - y^2_i + x^2_1\right) \vee \left(y^1_i - y^3_i + x^3_1\right) \right] \\ & = -5 \vee [((-1-2) \vee (-1-6)) \wedge ((-1-5) \vee (1-7)) \wedge ((-3+1) \vee (-3+5))] \\ & = -5 \vee [(-3) \wedge (-4) \wedge 2] \\ & = -4 \geq \tilde{x}^1_1 \end{aligned} \tag{5.44}$$

Also, if i = 1, then for the column index j = 3 we have

$$\begin{aligned}
&x_3^1 \vee \left(\bigvee_{\xi=1}^{3} [y_1^1 - y_1^\xi + x_3^\xi] \right) \\
&= x_1^1 \vee \left[\left(y_1^1 - y_1^2 + x_3^2\right) \vee \left(y_1^1 - y_1^3 + x_3^3\right) \right] \\
&= -1 \vee [((-1-2+0) \vee (-1-5+0))] \qquad (5.45) \\
&= -1 \vee [(-3) \vee (-7)] \\
&= -1 = \tilde{x}_3^1
\end{aligned}$$

A similar comparison shows that for i = 2, the column index j = 1 satisfies the equality $\tilde{x}_1^1 = x_1^1 \vee \left(\bigvee_{\xi=2}^{3} [y_3^1 - y_3^\xi + x_3^\xi]\right)$, and for i = 3, the column index j = 3 satisfies the equality $\tilde{x}_3^1 = x_3^1 \vee \left(\bigvee_{\xi=2}^{3} [y_3^1 - y_3^\xi + x_3^\xi]\right)$ Therefore, the pattern $\tilde{\mathbf{x}}^1$ satisfies the two conditions of Theorem 3 given by Equations (5.41) and (5.42). In this particular case, the memory W_{XY} is given by

$$W_{XY} = \begin{pmatrix} 2 & 5 & 0 \\ 5 & 7 & 2 \\ -1 & 0 & -2 \end{pmatrix} \qquad (5.46)$$

and

$$\begin{aligned}
W \boxtimes \tilde{\mathbf{x}}^1 &= \begin{pmatrix} 2 & 5 & 0 \\ 5 & 7 & 2 \\ -1 & 0 & -2 \end{pmatrix} \boxtimes \begin{pmatrix} -4 \\ -8 \\ -1 \end{pmatrix} \\
&= \begin{pmatrix} (-2) \vee (-3) \vee (-1) \\ 1 \vee (-1) \vee 1 \\ (-5) \vee (-8) \vee (-3) \end{pmatrix} \qquad (5.47) \\
&= \begin{pmatrix} -1 \\ 1 \\ -3 \end{pmatrix} = \mathbf{y}^1
\end{aligned}$$

which is in agreement with Theorem 3.

In the above example, $x_1^1 < \tilde{x}_1^1$ and, therefore, $\tilde{\mathbf{x}}^1 \not\leq \mathbf{x}^1$. However, perfect recall is still achieved. We shall reflect on this observation in our subsequent discussion on perfect recall of corrupted input patterns.

5.2. Autoassociative Morphological Memories

If $X = Y$ (i.e., $\mathbf{y}^\xi = \mathbf{x}^\xi$, for $\xi = 1, \cdots, k$), then W_{XX} and M_{XX} are called *autoassociative memories*. The classical example of a semilinear autoassociative memory is the Hopfield net [Hop82a,Lip87]. As shown in Section 5.1, the Hopfield net is given by

$$w_{ij} = \begin{cases} \sum_{\xi=1}^{k} x_i^\xi \cdot x_j^\xi & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases}. \qquad (5.48)$$

For an input pattern $\mathbf{x} = (x_1, \ldots, x_n)$, the Hopfield net algorithm proceeds by initializing $x_j(0) = x_j$ at time $t = 0$ and then iterates the recursive formula $x_j(t+1) = f[\sum_{i=1}^{n} w_{ij} x_i(t)]$ until convergence. The function f is a hard limiting nonlinearity.

The morphological analogues of the Hopfield memory are derived from Equation (5.23) and given by

$$w_{ij} = \bigwedge_{\xi=1}^{k} (x_i^\xi - x_j^\xi) \quad \text{and} \quad m_{ij} = \bigvee_{\xi=1}^{k} (x_i^\xi - x_j^\xi) \,. \tag{5.49}$$

As an example, consider the five pattern images $\mathbf{p}^1, \ldots, \mathbf{p}^5$ shown in the top row of Figure 5. 2. Each $\mathbf{p}^\xi$ is an 18x18-pixel Boolean image. Using the standard row-major scanning method, each pattern image $\mathbf{p}^\xi$ can be converted into a pattern vector $\mathbf{x}^\xi = (x_1^\xi, \cdots, x_{324}^\xi)$ by defining

$$x_{18(i-1)+j}^\xi = \begin{cases} 1 & \text{if } \mathbf{p}^\xi(i,j) = 1 \ (\textit{black pixel}) \\ 0 & \text{if } \mathbf{p}^\xi(i,j) = 0 \ (\textit{white pixel}) \end{cases} \,. \tag{5.50}$$

In this case, we obtain the perfect recall $W_{XX} \boxtimes \mathbf{x}^\xi = \mathbf{x}^\xi$ and $M_{XX} \boxtimes \mathbf{x}^\xi = \mathbf{x}^\xi$ for $\xi = 1, \ldots, 5$. However, using a Hopfield memory for these patterns does not result in perfect recall even in a nondistorted pattern is presented to the memory. The bottom row of Figure 5.2 shows the output of the Hopfield net when presented with the patterns representing the letters. Although the patterns "A" and "X" resulted in perfect recall, the patterns "B", "C", and "E" all converged to a configuration not represented by any of these patterns. The reason for this is that these three patterns are not very orthogonal to each other; i.e., there is a large subset of nonzero pixels common to all three patterns. It is well known that Hopfield memories, which are *linear* combinations of pattern features, have difficulties in memorizing patterns that share too many bits [Lip87].

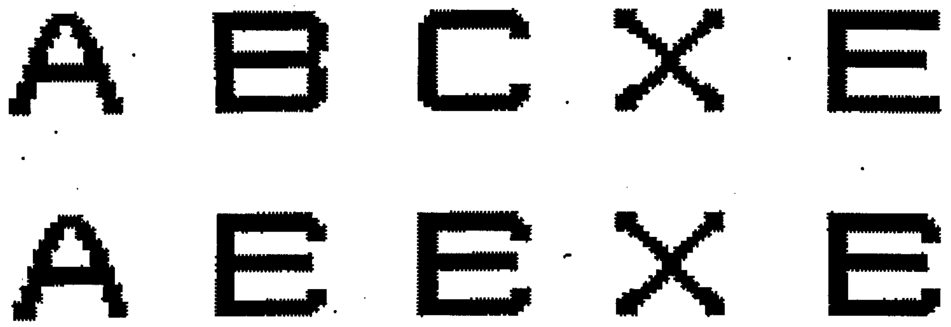


Figure 5.2. *The five patterns in the top row were used in constructing the Hopfield memory and the morphological memory. The output for either autoassociative morphological memory W_{XX} or M_{XX} is identical to the input patterns. The bottom row represents the output patterns of the Hopfield net when presented with the respective patterns in the top row.*

Although some improvements could be obtained by using certain orthogonalization procedures [Wal85,Gra86], an even more severe limitation for the Hopfield memory is the number of patterns that can be stored for accurate recall. For instance, doubling the number of patterns from five to ten by introducing the lower case letters shown in Figure 5.3 results in a complete recall failure. Figure 5.4 shows the output from the Hopfield memory when presented with the patterns shown in Figure 5.3.

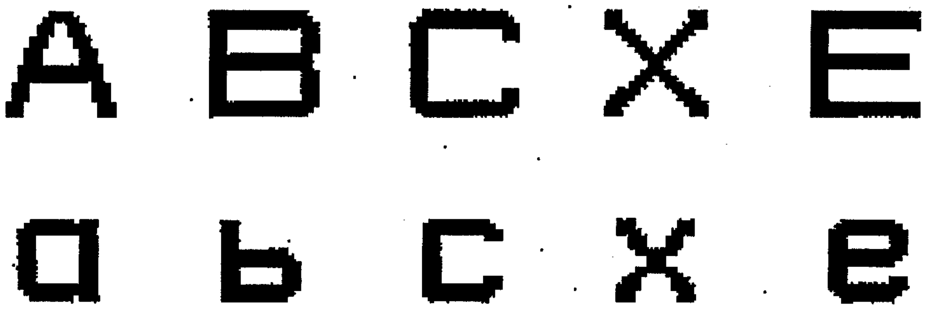


Figure 5.3. *The ten patterns used in constructing the Hopfield memory and the morphological memories. The output of either autoassociative morphological memory W_{XX} or M_{XX} is identical to the input patterns.*

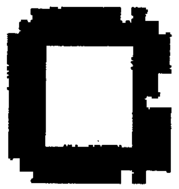


Figure 5.4. *The output of the Hopfield memory, which remains the same no matter which one of the patterns shown in Figure 5.3 is presented to the memory.*

5.2.1. Perfect Recall. In contrast to the Hopfield memory, in the absence of noise a natural morphological autoassociative memory will always provide perfect recall for *any* number of patterns programmed into its memory! The fact that morphological autoassociative memories have unlimited storage capacity is given by the next theorem.

Theorem 4. $W_{XX} \boxed{\vee} X = X$ *and* $M_{XX} \boxed{\wedge} X = X$.

Note that the theorem places no restriction on the type or number of patterns. In particular, the natural morphological memories give perfect recall for all ten patterns or, if desired, the entire Roman alphabet of upper- and lower-case letters. Additionally, a consequence of Theorem 4 is that the memories W_{XX} and M_{XX} are stable with respect to the input vectors $\mathbf{x}^{\xi}$. In fact, as the next theorem shows, the memories converge in *one step* for any input pattern $\mathbf{z}$.

Theorem 5. *If* $W_{XX} \boxed{\vee} \mathbf{z} = \mathbf{v}$ *and* $M_{XX} \boxed{\wedge} \mathbf{z} = \mathbf{u}$, *then* $W_{XX} \boxed{\vee} \mathbf{v} = \mathbf{v}$ *and* $M_{XX} \boxed{\wedge} \mathbf{u} = \mathbf{u}$.

Theorems 4 and 5 point out another major difference between Hopfield memories and morphological autoassociative memories. Not only is there an immense difference in storage capacity, but perfect recall is guaranteed and recall occurs in one step that can be implemented in parallel. In contrast, Hopfield memories must be run iteratively with neurons firing randomly, one at a time, and there is no guarantee of perfect recall for perfect inputs. Thus, morphological associative memories behave more like human associative memories. Once a pattern has been memorized, recall is instantaneous when the human brain is presented with the pattern.

5.2.2. Recall of Corrupted and Occluded Patterns. Since the Hopfield net is not capable of recalling a large number of perfect exemplar patterns, the Hopfield net's performance is even further degraded when these patterns are corrupted by noise or occluded by foreground artifacts. In contrast, the autoassociative morphological memories discussed thus far are extremely robust to certain types of noise and occlusions.

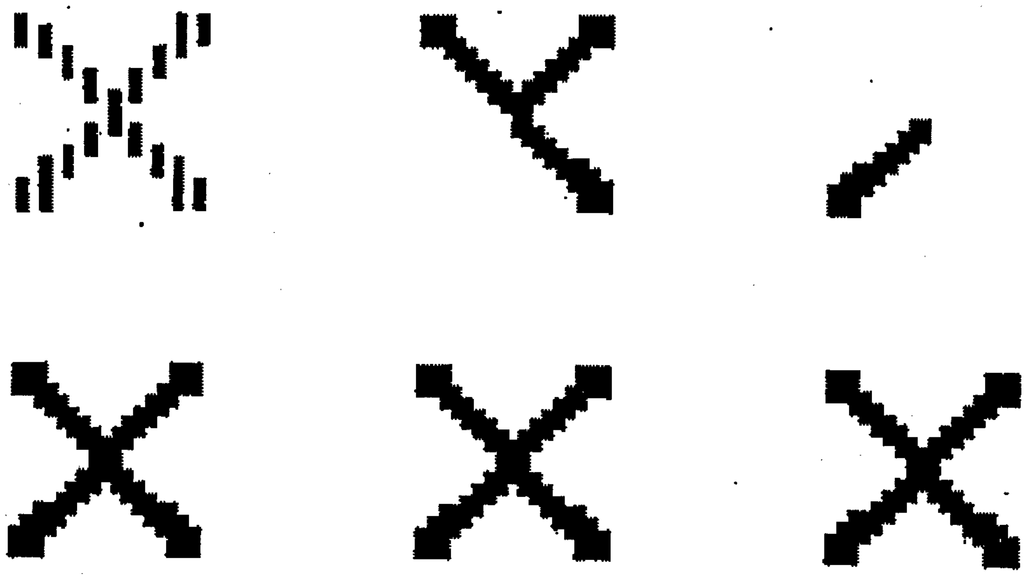


Figure 5.5. *The top row shows the corrupted input patterns and the bottom row, the corresponding output patterns of the morphological memory W_{XX}.*

We say that the distorted version $\tilde{\mathbf{x}}^\gamma$ of the pattern $\mathbf{x}^\gamma$ has undergone an *erosive change* whenever $\tilde{\mathbf{x}}^\gamma \leq \mathbf{x}^\gamma$ and a *dilative change* whenever $\tilde{\mathbf{x}}^\gamma \geq \mathbf{x}^\gamma$. Thus, for the above Boolean patterns, a change in pattern values from $\mathbf{p}(i,j) = 1$ to $\mathbf{p}(i,j) = 0$ represents an erosive change, while a change from $\mathbf{p}(i,j) = 0$ to $\mathbf{p}(i,j) = 1$ represents an dilative change. The morphological autoassociative memory W_{XX} is extremely robust in recalling patterns that are distorted due to erosive changes. These changes can be random (system noise, partial pattern occlusion, etc.) or nonrandom (processing effects such as skeletonizing, filtering, etc.) The limitations on the amount of distortions are given by the theorems stated in the preceding section. For example, defining W_{XX} in terms of the ten patterns shown in Figure 5.3, and corrupting any of the patterns with 50 percent randomly generated erosive noise (i.e., black pattern pixels turned white randomly, pixel by pixel) always resulted in perfect recall. Figure 5.5 shows three experimental examples where the pattern represented by the letter "X" was intentionally corrupted by use of large erosive changes. In contrast to the Hopfield net, which failed to recall the exemplar pattern in each case, the morphological autoassociative memory provides perfect recall in all three cases.

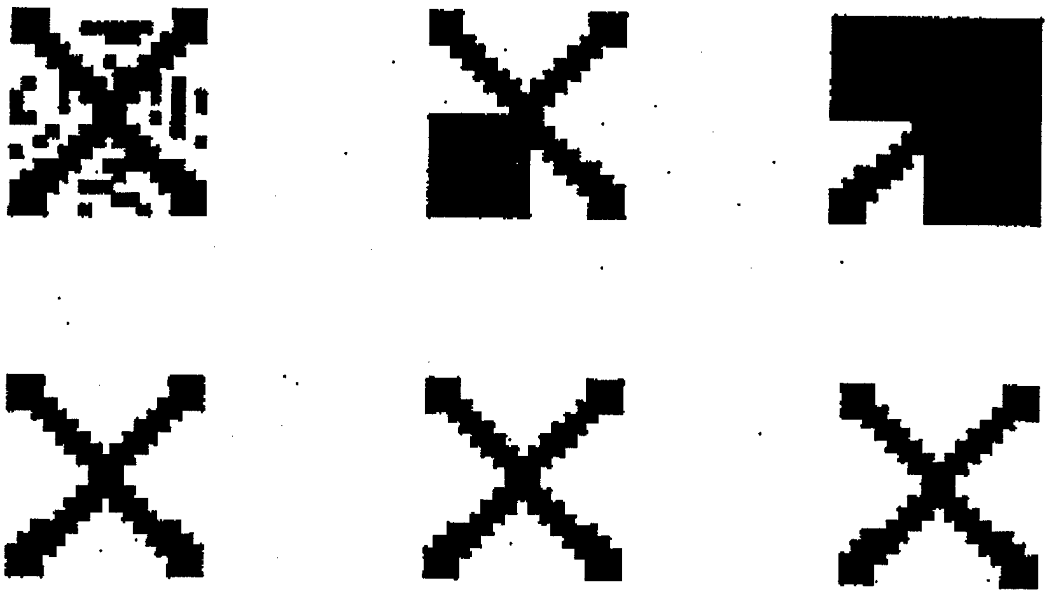


Figure 5.6. *The top row shows the corrupted input patterns and the bottom row, the corresponding output patterns of the morphological memory M_{XX}.*

Although the performance of the autoassociative morphological memory W_{XX} is excellent in the presence of erosive noise, pattern recall becomes impossible when dilative noise in introduced. For dilative noise, the dual memory M_{XX} proves to be the ideal associative memory. This follows from Theorem 4 and is also supported by experiment. For example, defining M_{XX} in terms of the ten patterns shown in Figure 5.3 provides for an associative memory that is extremely robust in the presence of dilative noise of these patterns. For instance, the top row of Figure 5.6 shows several version of the letter "X" corrupted by severe dilative changes. In each of these cases, the memory M_{XX} provided perfect recall.

The problem with the associative memories W_{XX} and M_{XX} is that W_{XX} is incapable of effectively handling dilative noise while M_{XX} is incapable of effectively handling erosive noise. A consequence of Theorem 3 is that if $\tilde{\mathbf{x}}^{\gamma} > \mathbf{x}^{\gamma}$, then in many instances, $\left(W_{XX} \boxed{\vee}\ \tilde{\mathbf{x}}^{\gamma}\right)_i \neq y_i^{\xi}$, and if $\tilde{\mathbf{x}}^{\gamma} > \mathbf{x}^{\gamma}$, then we have a high probability that $\left(M_{XX} \boxed{\wedge}\ \tilde{\mathbf{x}}^{\gamma}\right)_i \neq y_i^{\xi}$. Hence, dilative noise destroys perfect recall for W_{XX} while erosive noise destroys recall for M_{XX}. However, we need to recall that Example 3 shows that a certain amount of dilative noise may not destroy perfect recall for W_{XX}. A similar observation holds for erosive noise and M_{XX}.

5.3. Perfect Recall Conditions for Distorted Inputs

Although theorems provide necessary and sufficient bounds for the class of inputs that will result in perfect recall, they also point out the extreme vulnerability of the morphological memories W_{XX} and M_{XX} if patterns exhibit both erosive and dilative noise. For this reason, the vector

$$W_{XX} \boxed{\vee} \left(M_{XX} \boxed{\wedge}\ \tilde{\mathbf{x}}^{\gamma}\right) \tag{5.51}$$

will generally not correspond to $\mathbf{x}^{\gamma}$ if $\tilde{\mathbf{x}}^{\gamma}$ corresponds to a corrupted version of $\mathbf{x}^{\gamma}$ containing both erosive and dilative noise. In fact, since $\tilde{\mathbf{x}}^{\gamma}$ contains erosive noise, the expression

$M_{XX} \boxed{\vee} \tilde{\mathbf{x}}^{\gamma}$ will most likely correspond to the zero vector. However, the naïve idea of first presenting a noisy pattern to a memory that is insensitive to dilative noise followed by presenting the output pattern to a memory that is insensitive to erosive noise, as represented by Equation (5.51), does furnish a first step in constructing an associative morphological memory that is robust in the presence of random noise, random occlusion, as well as being capable of recalling either skeletonized or dilated versions of patterns.

The fundamental idea of creating a morphological associative recall memory that is robust in the presence of random noise (i.e., both dilative and erosive random noise) is to replace the associative memory M_{XY} with a sequence of two memories M_1 and W_1 that are defined in terms of subpatterns of the exemplar patterns $\mathbf{x}^{\xi}$. Basically, the memory M_1 is constructed such that it can associate randomly corrupted version of the exemplar $\mathbf{x}^{\xi}$ with a subpattern $\mathbf{z}^{\xi}$ consisting of a few select pattern values from $\mathbf{x}^{\xi}$. The pattern $\mathbf{z}^{\xi}$ represents a special eroded version of $\mathbf{x}^{\xi}$. In the Boolean case, ideally, $\mathbf{z}^{\xi}$ is a sparse representation of $\mathbf{x}^{\xi}$ having the property that $\mathbf{z}^{\xi} << \mathbf{x}^{\xi}$ and $\mathbf{z}^{\xi} \wedge \mathbf{x}^{\gamma} = 0$ whenever $\xi \neq \gamma$. Here << denotes "much smaller than" in the sense of containing mostly zero-valued pixels , and 0 denotes the zero vector. The morphological memory M_1 is defined as an associative memory for the patterns $\mathbf{x}^{\xi}$, $\xi = 1, \cdots, k$ so that $M_1 \boxed{\vee} \tilde{\mathbf{x}}^{\xi} = \mathbf{z}^{\xi}$. Furthermore, since $\mathbf{x}^{\xi}$ represents a version of $\mathbf{z}^{\xi}$ corrupted only by dilative noise, we also have $M_1 \boxed{\vee} \mathbf{x}^{\xi} = \mathbf{z}^{\xi}$.

The memory W_1 is defined as an associative memory that associates with each input $\mathbf{x}^{\xi}$ the pattern $\mathbf{y}^{\xi}$. Thus, under the correct conditions, we obtain $W_1 \boxed{\wedge} \left(M_1 \boxed{\vee} \mathbf{x}^{\xi}\right) = W_1 \boxed{\wedge} \mathbf{z}^{\xi} = \mathbf{y}^{\xi}$ as well as $W_1 \boxed{\wedge} \left(M_1 \boxed{\vee} \tilde{\mathbf{x}}^{\xi}\right) = W_1 \boxed{\wedge} \mathbf{z}^{\xi} = \mathbf{y}^{\xi}$ for randomly corrupted patterns $\mathbf{x}^{\xi}$.

We now make these notions more rigorous.

Definition 2. Let $Z = (\mathbf{z}^1, \mathbf{z}^2, \cdots, \mathbf{z}^k)$ be an nxk matrix. We say that Z is a *kernel* for (X,Y) if and only if the following two conditions are satisfied:

1. $M_{ZZ} \boxed{\vee} X = Z$;
2. $W_{ZY} \boxed{\wedge} Z = Y$.

It follows that if Z is a *kernel* for (X,Y), then

$$W_{ZY} \boxed{\wedge} \left(M_{ZZ} \boxed{\vee} X\right) = W_{ZY} \boxed{\wedge} Z = Y. \tag{5.52}$$

Equation (5.52) defines the morphological associative memory system

$$\textit{input} \rightarrow M_{ZZ} \rightarrow W_{XY} \rightarrow \textit{output} \tag{5.53}$$

which, for a properly chosen kernel Z, will be robust in the presence of random noise in input X.

Definition 2 does not provide for a methodology of finding kernels nor does it furnish direct insight into the question of why kernels are useful in constructing associative memories that are robust in the presence of random noise. The next theorem provides a partial answer to these questions.

Theorem 6. *If Z is a kernel for (X,Y), then $Z \leq X$.*

If $Z = (\mathbf{z}^1, \mathbf{z}^2, \cdots, \mathbf{z}^k)$ is a kernel for (X,Y), then the column vectors $\mathbf{x}^{\xi}$ of Z are called *kernel vectors* for (X,Y). According to Theorem 6, $\mathbf{z}^{\xi} \leq \mathbf{x}^{\xi}$ for ξ = 1, 2, …, k. Thus, the kernel vectors represent eroded versions of the patterns $\mathbf{x}^1$, …, $\mathbf{x}^k$.

Example 4. The set

$$\left\{\begin{pmatrix}0\\0\\-2\end{pmatrix},\begin{pmatrix}0\\-2\\-4\end{pmatrix},\begin{pmatrix}0\\-3\\-2\end{pmatrix}\right\} \tag{5.54}$$

is a set of kernel vectors for the set of vector pairs given in Example 1. The corresponding memories M_{ZZ} and W_{ZY} are given by

$$M_{ZZ}=\begin{pmatrix}0&3&4\\0&0&2\\-2&1&0\end{pmatrix} \quad \text{and} \quad W_{ZY}=\begin{pmatrix}-1&0&2\\-1&1&0\\0&0&2\end{pmatrix}. \tag{5.55}$$

The set

$$\left\{\begin{pmatrix}0\\0\\-3\end{pmatrix},\begin{pmatrix}0\\-2\\-4\end{pmatrix},\begin{pmatrix}0\\-3\\-3\end{pmatrix}\right\} \tag{5.56}$$

provides another example of a set of kernel vectors for the vector pairs given in Example 1. This shows that kernel vectors are not unique.

The definition of kernel vectors does not preclude that $\mathbf{z}^\xi = \mathbf{x}^\xi$ for some ξ or even that $Z=X$. of course, if $Z=X$, then nothing will have been gained in the quest for creating morphological associative memories that are robust in the presence of random noise. For the same reason, cases where $\mathbf{z}^\xi = \mathbf{x}^\xi$ for some ξ need to be avoided. This leads to the notion of *proper kernels*. If Z is a kernel for (X,Y) with the property that $\mathbf{z}^\xi \neq \mathbf{x}^\xi \ \forall\xi$, then Z is called a *proper kernel* for (X,Y). Note that the two kernels in Example 4 are not proper kernels since for both kernels, we have $\mathbf{z}^2 = \mathbf{x}^2$. However, the notion of a proper kernel does not imply that $\mathbf{z}^\xi < \mathbf{x}^\xi \ \forall\xi$. It only implies that for each $\xi = 1, 2, \ldots, k$, there exists an index $i_\xi \in \{1,\cdots,n\}$ such that $z^\xi_{i_\xi} \neq x^\xi_{i_\xi}$. In other words, each kernel vector $\mathbf{z}^\xi$ has *at least one* coordinate that differs from the corresponding coordinate of $\mathbf{x}^\xi$. According to Theorem 6, this coordinate must be *strictly less than* the corresponding coordinate of $\mathbf{x}^\xi$.

If $\mathbf{z}^\gamma \le \tilde{\mathbf{x}}^\gamma \le \mathbf{x}^\gamma$, then

$$\mathbf{z}^\gamma = M_{ZZ} \boxed{\wedge} \mathbf{z}^\gamma \le M_{ZZ} \boxed{\wedge} \tilde{\mathbf{x}}^\gamma \le M_{ZZ} \boxed{\wedge} \mathbf{x}^\gamma = \mathbf{z}^\gamma \tag{5.57}$$

and, hence, $M_{ZZ} \boxed{\wedge} \tilde{\mathbf{x}}^\gamma = \mathbf{z}^\gamma$. Thus, if Z is a kernel for (X,Y) and $\mathbf{z}^\gamma \le \tilde{\mathbf{x}}^\gamma \le \mathbf{x}^\gamma$, then we are guaranteed that

$$W_{ZY} \boxed{\vee} \left(M_{ZZ} \boxed{\wedge} \tilde{\mathbf{x}}^\gamma\right) = \mathbf{y}^\gamma. \tag{5.58}$$

Therefore, an eroded version $\tilde{\mathbf{x}}^\gamma$ of $\mathbf{x}^\gamma$ satisfying the inequality $\mathbf{z}^\gamma \le \tilde{\mathbf{x}}^\gamma$ will be correctly associated with the pattern $\mathbf{y}^\gamma$. Since M_{ZZ} is very robust in the presence of dilative noise, it is not unreasonable to expect that a distorted version $\tilde{\mathbf{x}}^\gamma$ of $\mathbf{x}^\gamma$ containing both dilative and erosive noise while satisfying the inequality $\mathbf{z}^\gamma \le \tilde{\mathbf{x}}^\gamma$ will be correctly associated with the pattern $\mathbf{y}^\gamma$. This expectation turns out to be true and the next theorem provides conditions that guarantee perfect recall for corrupted input patterns $\tilde{\mathbf{x}}^\gamma$.

Theorem 7. *If Z is a kernel for (X,Y) and* $\tilde{\mathbf{x}}^{\gamma}$ *denotes a distorted version of* $\mathbf{x}^{\gamma}$ *such that* $z_j^{\gamma} \leq \tilde{x}_j \; \forall j = 1, \cdots, n$ *and* $z_j^{\gamma} = \tilde{x}_j$ *whenever* $m_{ZZ}(i,j) = z_i^{\gamma} - z_j^{\gamma}$, *then* $M_{ZZ} \boxed{\wedge} \tilde{\mathbf{x}}^{\gamma} = \mathbf{z}^{\gamma}$.

In this theorem, $m_{ZZ}(i,j)$ denotes the i,jth entry of the matrix M_{ZZ}. The theorem also shows that any corrupted version $\tilde{\mathbf{x}}^{\gamma}$ of $\mathbf{x}^{\gamma}$ which contains both dilative and erosive noise but satisfies the hypothesis will be correctly associated with the pattern $\mathbf{y}^{\gamma}$ when fed into the system $\{input \rightarrow M_{ZZ} \rightarrow W_{ZY} \rightarrow output\}$.

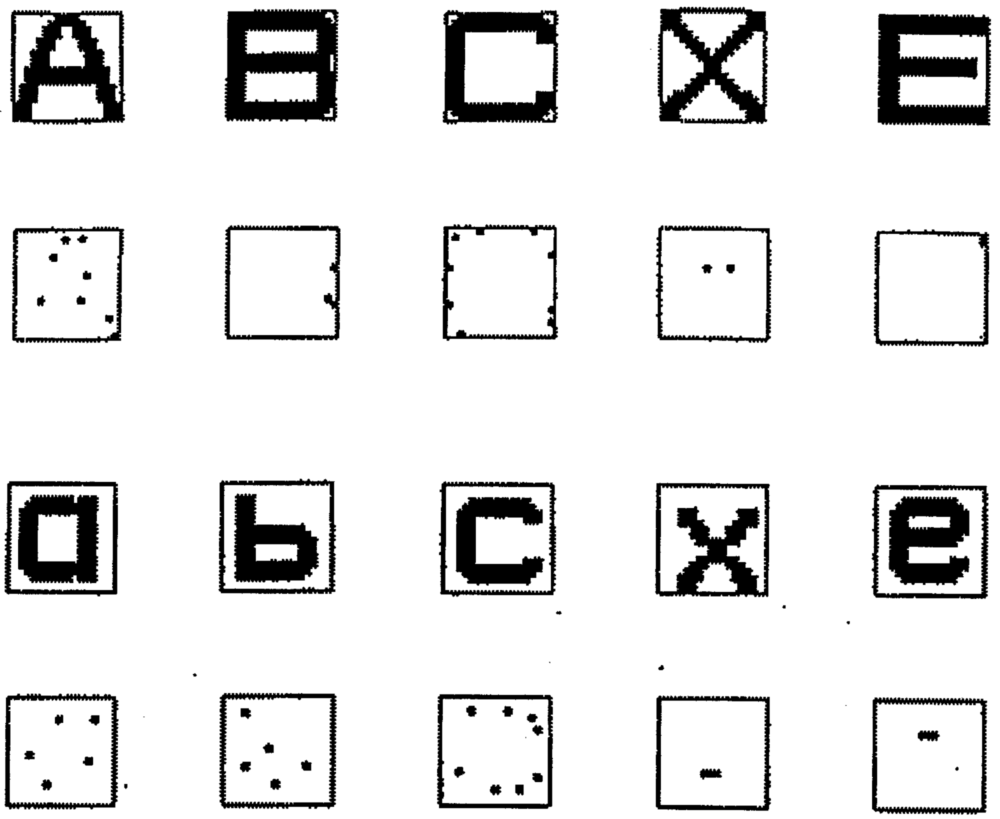


Figure 5.7: *An example of kernel images. The kernel image corresponding to a particular letter image is the image directly below the letter image.*

Theorems 6 and 7 can be viewed as a guide for selecting kernel vectors. A first step is to erode the patterns $\mathbf{x}^{\xi}$, saving only a few unique pattern values in order to construct a kernel vector $\mathbf{z}^{\xi}$ such that $\mathbf{z}^{\xi} \wedge \mathbf{x}^{\gamma} = 0$ whenever $\gamma \neq \xi$. This may not always be possible. However, when it is possible, it provides unique identification markers for the memory M_{ZZ} and the conditions stated in Definition 2 are automatically satisfied. In general, however, such unique features may not exist for some index ξ. In this case, once an initial set of kernel vector candidates has been selected, the two conditions of Definition 2 need to be checked in order to ensure correct recall.

We employed this method for obtaining kernel vectors $\mathbf{x}^{\xi}$ and corresponding weight matrices M_{ZZ} and W_{ZX} for the ten patterns shown in Figure 5.3. The corresponding kernel images are shown in Figure 5.7. The autoassociative memory $\{input \rightarrow M_{ZZ} \rightarrow W_{ZY} \rightarrow output\}$ obtained from this set of kernel vectors proved to be extremely roust, recognizing all patterns even after each pattern was corrupted by randomly reversing each bit with a probability of 0.15. Figures 5.8 and 5.9 represent a sampling of our inputs to the net and the corresponding outputs.

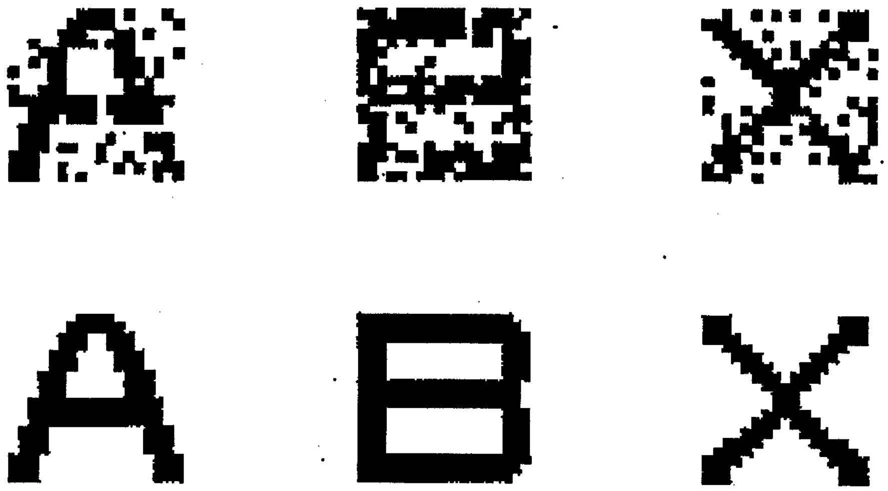


Figure 5.8. *An example of the behavior of the memory* $\{input \rightarrow M_{ZZ} \rightarrow W_{ZY} \rightarrow output\}$, *which was trained using the ten exemplars shown in Figure 5.3. Presenting the memory with the corrupted patterns of the letters A, B, and X resulted in perfect recall (lower row). Each letter was corrupted by randomly reversing each bit with a probability of 0.15.*

An additional benefit of using kernel patterns is the increase in storage capacity for associations (X,Y). This is due to the fact that M_{ZZ} I a perfect recall memory for any finite number of pattern vectors $\mathbf{z}^{\xi}$ and because of the sparseness of eroded patterns (i.e., in the Boolean case most pattern values will be zero), the storage capacity of W_{ZY} can be greatly increased.

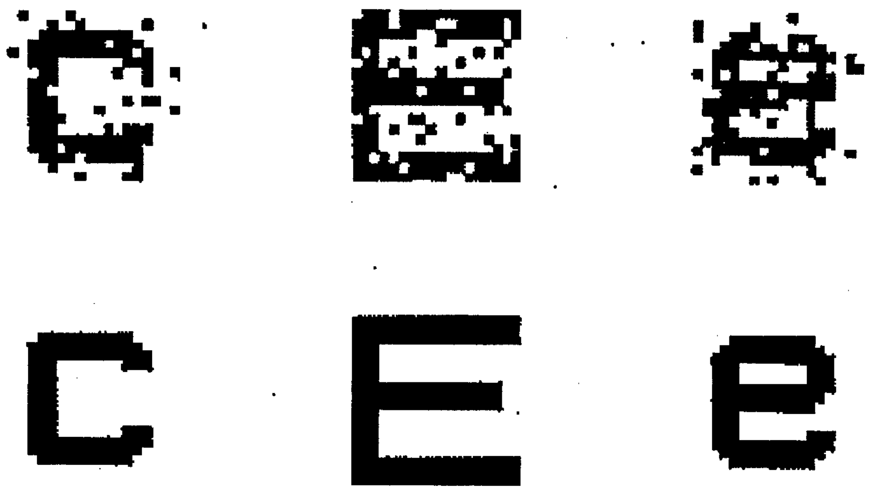


Figure 5.9. Another example of the behavior of the memory $\{input \rightarrow M_{ZZ} \rightarrow W_{ZY} \rightarrow output\}$. Presenting the memory with the corrupted patterns of the letters c, E, and e resulted in perfect recall (lower row). Again, each letter was corrupted by randomly reversing each bit with a probability of 0.15.

5.4. Comparison with Previous Work

In recent years, lattice-based matrix operations have found widespread applications in the engineering sciences. In these applications, the usual matrix operations of addition and multiplication are replaced by corresponding lattice operations. Lattice induced matrix operations lead to an entirely different perspective of a class of nonlinear transformations. These ideas were applied by Shimbel [Shi54] to communications networks, and to machine scheduling by Cuninghame-Green [Cun60,62] and Giffler [Gif60]. Others have discussed their usefulness in applications featuring shortest-path problems in graphs [Bac75,Ben68,Car71,Pet67]. Additional examples are given in [Cun79], primarily in the field of operations research.

The concept of morphological networks grew out of the theory of image algebra [Rit01,RitMS.Rit91,Rit90a]. It was shown that a subalgebra of image algebra includes the mathematical formulations of currently popular neural network models [Rit89,Rit91] and first attempts in formulating useful morphological neural networks appeared in [Rit90b] and [Dav90]. Applications to image processing were first developed by Ritter and Davidson [Rit90b,Dav89]. Davidson employed morphological neural networks to solve template identification and target classification problems [Dav92b,93a-c]. C.P. Suarez-Araujo applied morphological neural nets to compute homothetic auditory and visual invariances [Sua92,97]. Another interesting network consisting of a morphological net and a classical feedforward network used for feature extraction and classification was designed by Paul Gader and his colleagues [Gad94,Won95]. These researchers devised multilayer morphological neural nets for very specialized applications. A more comprehensive and rigorous basis for computing with MNNs appeared in [Rit96] where it was shown that morphological neural nets are capable of solving any conventional computational problem. Ritter, Sussner, and Diaz-de-Leon [Rit98] investigated morphological associative memories, a class of networks not previously discussed in detail.

An entirely different model of a morphological network was presented in [Wil89]. This particular model employs the customary operations of multiplication and summation at each node, which is fundamentally different from the models described herein.

5.5. Applications and Future Work

We have developed a new framework of neural network computing based on lattice algebra. The resulting morphological neural networks are radically different in behavior than traditional artificial neural network models. The main emphasis of this paper was on morphological associative memories. In contrast to traditional associative memories, morphological associative memories converge in one step! Thus, convergence problems do not exist. Morphological analogues to the Hopfield net not only proved to be far more robust in the presence of noise, but also have unlimited storage capacity for perfect inputs. For noisy inputs and carefully chosen kernels, morphological autoassociative memories again exhibit superior performance in both recall and storage. In our experiment, the autoassociative morphological memory $\{input \rightarrow M_{ZZ} \rightarrow W_{ZY} \rightarrow output\}$ was capable of recalling 62 grid patterns defined on grids of size 21x21 pixels. These patterns represent the entire English alphabet of upper- and lower-case letters together with integer digits ranging from 0 to 9. Corruption of these patterns by randomly reversing each bit with a probability of 0.08 and presenting these corrupted patterns to the memory also resulted in perfect recall.

It should be emphasized that the results presented herein represent only a first step in the exploration of morphological neural networks. The lattice algebraic approach to neural network theory is new and a multitude of open questions await exploration. For example, we noted that the selection of kernel patterns merits further investigation. This is especially true when using gray-valued patterns. Selection of proper sets of kernel vectors for Boolean patterns is usually easy, while the selection of proper sets for gray-valued kernel patterns can be difficult. To obtain a better understanding of this difficulty, we challenge the reader to find a proper set of kernel vectors for the set of vector pairs defined in Example 1. Of course, this represents only a small part of the overall challenge. Learning rules for both associative morphological memories and morphological perceptrons are under development, but need to be further investigated. The base of applications needs to be expanded and further compared to traditional methods. Dr. Ritter and his students are currently investigating the use of single neurons for computation, including applications in target recognition. Early results have shown that morphological models for single neurons with dendritic computation can be employed in target classification (e.g., landmine detection applications) with very high probability of detection and low false alarm rate.

In a larger sense, MNN models are connected with the fundamental question concerning the difference between biological neural networks and artificial neural networks, namely,

Is the strength of the electric potential of a signal traveling along an axon the result of a multiplicative process, and does the mechanism of a neuron's postsynaptic membrane add the various potentials of electrical impulses, or is the strength of the electric potential an additive process and does the postsynaptic membrane only accept signals of a certain maximum strength?

A positive answer to the latter query would provide a strong biological basis for morphological neural networks. In Section 6, we discuss the application of MNNs to processing of the TNE agreement map.

5.6. References

[Abu85] Abu-Mostafa, Y. and J. St. Jacques. "Information capacity of the Hopfield model", *IEEE Transactions on Information Theory* **IT-7**:1-11 (1985).

[Ami85] Amit, D. , H. Gutfruend, and H. Sompolinsky. "Storing infinite number of patterns in a spin-glass model neural network", *Physics Review Letters* **55**(14):1530-1533 (1985).

[Bac75] Backhouse, R.C. and B. Carre. "Regular algebra applied to path-finding problems", *J. Inst. Math. Applicat.* **15**:161-186 (1975).

[Ben68] Benzaken, C. "Structures algebra des cheminements", in *Network and Switching Theory*, G. Biorci, ed., New York: Academic Press, pp. 40-57 (1968).

[Car71] Carre, B. "An algebra for network routing problems", *J. Inst. Math. Applicat.* **7**:273-294 (1971).

[Che86] Chen, H. et al. "Higher order correlation model for associative memories", in *Neural Networks for Computing*, J.S. Denker, Ed., *AIP Proceedings* **151** (1986).

[Cun60] Cuninghame-Green, R. "Process synchronization in steelworks – A problem of feasibility", in *Proceedings of the Second International Conference on Operations Research*, A. Banbury and D. Maitland, Eds., London: English University Press, pp. 323-328 (1960).

[Cun62] Cuninghame-Green, R. "Describing industrial processes with interference and approximating their stead-state behavior", *Operations Research Quarterly* **13**:95-100 (1962).

[Cun79] Cuninghame-Green, R. *Minimax Algebra*: *Lecture Notes in economics and Math. Syst.*, Volume **166**, New York: Springer-Verlag (1979).

[Dav89] Davidson, J.L. *Lattice Structures in the Image Algebra and Applications to Image Processing*, Ph.D. Dissertation, University of Florida, Gainesville FL (1989).

[Dav90] Davidson, J.L. and G.X. Ritter. "A theory of morphological neural networks", *Proceedings SPIE* **1215**:378-388 (1990).

[Dav92a] Davidson, J.L. "Foundations and Applications of Lattice Transforms in Image Processing", in *Advances in Electronics and Electron Physics* (P. Hawkes, Ed.) **84**:61-130 (1992).

[Dav92b] Davidson, J.L. "Simulated annealing and morphological neural networks", *Proceedings SPIE* **1769**:119-127 (1992).

[Dav93a] Davidson, J.L. and F. Hummer. "Morphology neural networks: An introduction with applications", *IEEE Syst. Signal Processing* **12**:177-210 (1993).

[Dav93b] Davidson, J.L. and R. Srivastava. "Fuzzy image algebra neural network for template identification", *Proceedings of the IEEE Second Annual Midwest Electrotechnology Conference* , pp. 68-71 (1993).

[Dav93c] Davidson, J.L. and A. Talukder. "Template identification using simulated annealing in morphology neural networks", *Proceedings of the IEEE Second Annual Midwest Electrotechnology Conference* , pp. 64-67 (1993).

[Den86] Denker, J.S. "Neural-network models of learning and adaptation", *Physica D* **22**:216-222 (1986).

[Gad94] Gader, P.D., Y. Won, and M.A. Khabou. "Image algebra network for pattern recognition", in *Proceedings SPIE* **2300**:157-168 (1994).

[Gra86] Grant, P.M. and J.P. Sage. "A comparison of neural network and matched filter processing for detecting lines in images", in *AIP Conference Proceedings* **151**, *Neural Networks for Computing*, J.S. Denker, Ed. (1986).

[Gif60] Giffler, B. "Mathematical solution of production planning and scheduling problems", IMB ASDD, Technical Report (1960).

[Hei94] Heijmans, H.J.A.M. *Morphological Image Operators*, Boston, MA: Academic Press (1994).

[Hop82a] Hopfield, J.J. "Neural networks and physical systems with emergent collective computational abilities", in *Proceedings of the National Academy of Sciences* **79**:2554-2558 (1982).

[Hop82b] Hopfield, J.J. "Neurons with graded response have collective computational properties like those of two-state neurons", in *Proceedings of the National Academy of Sciences* **81**:3088-3092 (1982).

[Hop86] Hopfield, J.J. and D.W. Tank. "Computing with neural circuits", *Science* **233**:625-633 (1986).

[Kee86] Keeler, J.D. "Basins of attraction of neural network models", in *Neural Networks for Computing*, J.S. Denker, Ed., *AIP Proceedings* **151** (1986).

[Koc92] Koch, C. and T. Poggio. "Multiplying with Synapses and Neurons", in *Single Neuron Computation*, (T. McKenna, J. Davis, S.F. Zornetzer, Eds.), San Diego, CA: Academic Press, pp. 315-345 (1992).

[Kos87a] Kosko, B. "Adaptive bidirectional associative memories", *Proceedings of the IEEE 16th Workshop on Appl. Images and Pattern Recognition*, pp. 1-49 (1987).

[Kos87b] Kosko, B. "Adaptive bidirectional associative memories", *IEEE Transactions on Systems, Man, and Cybernetics*, pp. 124-136 (1987).

[Lip87] Lippmann, R.P. "An introduction to computing with neural nets", *IEEE Transactions on Acoustics, Speech, and Signal Processing* **ASSP-4**:4-22 (1987).

[McE87] McEliece, R. et al. "The capacity of Hopfield associative memory", *IEEE Transactions on Information Theory* **IT-1**:33-45 (1987).

[Pet67] Peteanu, V. "An algebra of the optimal path in networks", *Mathematica* **9**:335-342 (1967).

[Pet98] Petridis, V. and V.G. Kaburlasos. "Fuzzy lattice neural network (FLNN): A hybrid model for learning", *IEEE Transactions on Neural Networks* **9**(5):877-890 (1998).

[Rit87] Ritter, G.X. and J.L. Davidson. "The image algebra and lattice theory", University of Florida CIS Department Technical Report TR-87-09 (1987).

[Rit89] Ritter, G.X., D. Li, and J.N. Wilson. "Image algebra and its relationships to neural networks", in *Proceedings of the SPIE Technical Symposium Southeast on optics, Electro-Optics, and Sensors, Orlando FL* (1989).

[Rit90a] Ritter, G.X., J.N. Wilson, and J.L. Davidson. "Image algebra: An overview", *Computer Vision, Graphics, and Image Processing* **49**(3):297-331 (1990).

[Rit90b] Ritter, G.X. and J.L. Davidson. "Recursion and feedback in image algebra", in *Proceedings of the SPIE 19th AIPR Workshop on Image Understanding in the 90s, McLean VA* (1990).

[Rit91] Ritter, G.X. "Recent developments in image algebra", *Advances in Electronics and Electron Physics* **80**:243-308 (1991).

[Rit96] Ritter, G.X. and P. Sussner. "An introduction to morphological neural networks", *Proceedings of the 13th International Conference on pattern Recognition (Vienna, Austria)*, pp.709-717 (1996).

[Rit97a] Ritter, G.X. and P. Sussner. "Associative memories based on lattice algebra", *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (Orlando, FL)*, pp. 3570-3575 (1997).

[Rit97b] Ritter, G.X. and P. Sussner. "Morphological perceptrons", in *Proceedings of ISAS '97 Conference on Intelligent Systems and Semiotics (Gaithersburg, MD)*, pp. 221-226 (1997).

[Rit98] Ritter, G.X., P. Sussner, and J.L. Diaz de Leon. "Morphological associative memories", *IEEE Transactions on Neural Networks* 9(2):281-293 (1998).

[Rit99a] Ritter, G.X. and T. Beavers. "An introduction to morphological perceptrons", in *Proceedings of the IEEE ANNIE '99 Conference on Artificial Neural Networks in Engineering (St. Louis, MO)* (1999).

[Rit99b] Ritter, J.L. Diaz de Leon, and P. Sussner. "Morphological bidirectional associative memories", *Neural Networks* **12**:851-867 (1999).

[Rit01] Ritter, G.X. and J.N. Wilson. *Handbook of Computer Vision Algorithms in Image Algebra*, Second Edition, Boca Raton, FL: CRC Press (2001).

[Rit02] Ritter, G.X. and G. Urcid. "Lattice algebra approach to single neuron computation", in *IEEE Transactions on Neural Networks* (Nov 2002).

[RitMS] Ritter, G.X. *Image Algebra with Applications*, Unpublished Manuscript, available via anonymous FTP at ftp://ftp.cise.ufl.edu/pub/src/ia/documents.

[RitXX] Ritter, G.X., G. Urcid, and L. Iancu. "Reconstruction of noisy patterns using morphological associative memories", to be published in *Journal of Mathematical Imaging and Vision.*

[Shi54] Shimbel, A. "Structure in communication nets", in *Proceedings of the Symposium on Information and Networks,* Polytechnic Institute of Brooklyn, pp. 119-203 (1954).

[Sua92] Suarez-Araujo, C.P. and G.X. Ritter. "Morphological neural networks and image algebra in artificial perception systems", in *Proceedings SPIE* **1769**:128-142 (1992).

[Sua97] Suarez-Araujo, C.P. "Novel neural network models for computing homothetic invariances: An image algebra notation", *Journal of Mathematical Imaging and Vision* 7(1):69-83 (1997).

[Sus00] Sussner, P. "Observations on morphological associative memories and the kernel method", *Elsevier Neurocomputing* **31**:167-183 (2000).

[Wal85] Wallace, D.J. "Memory and learning in a class of neural models", in *Proceedings of the Workshop on Lattice Gauge Theory*, B. Bunk and K.H. Mutter, Eds., Wuppertal, Germany: Plenum Press (1985).

[Wil89] Wilson, S.S. "Morphological networks", in *Proceedings of the SPIE Conference on Visual Communication and Image Processing IV*, Bellingham, WA: SPIE (1989).

[Won95] Won, Y. and P.D. Gader. "Morphological shared weight neural network for pattern classification and automatic target detection", in *Proceedings of the IEEE 1995 International Conference on Neural Networks* **4**:2134-2138 (1995).

6. Combination of MNNs and TNE in Classify-Before-Detect ATR

In this STTR, we established two distinct approaches to target classification using morphological neural networks to refine classifier output. Section 6.1 describes how MNNs can be employed in reconfiguration of the TNE agreement map (AM). Section 6.2 describes the application of Dr. Ritter's current research in feedforward MNNs and dendritic computation, and how these techniques can be used to enhance AM processing.

6.1. AM Processing with MNNs

Consider a signal vector $\mathbf{v}^{s(t)}$, which denotes output of a sensor s at time t, which is of form

$$\mathbf{v}^{s(t)} = \left(v_1^{s(t)}, v_2^{s(t)}, \ldots, v_i^{s(t)}, \ldots, v_n^{s(t)}\right)^{\mathrm{T}}, \tag{6.1}$$

where T denotes transpose and $v_i^{s(t)} \in \mathbf{N}$, with $\mathbf{N}$ denoting the set of positive integers. For each $I = 1, \ldots, n$, the ith component $v_i^{s(t)}$ of $\mathbf{v}^{s(t)}$ references a particular row of the TNE binary pointer table entry corresponding to the value $v_i^{s(t)}$. The AM table corresponding to $\mathbf{v}^{s(t)}$ is a new table consisting of n binary rows, where the ith row consists of the TNE binary pointer table entry corresponding to $v_i^{s(t)}$. Figure 6.1 illustrates the construction of the AM for $\mathbf{v}^{s(t)}$. Let $\mathbf{w}^{s(t)}$ denote the AM for $\mathbf{v}^{s(t)}$. Then, $\mathbf{w}^{s(t)}$ can be viewed as a vector, where $w_i^{s(t)}$ denotes the ith element of this vector, which is the ith row of $\mathbf{w}^{s(t)}$. Alternatively, $w_i^{s(t)}$ can be expressed as the integer value that corresponds to a binary representation of the ith row of the AM.

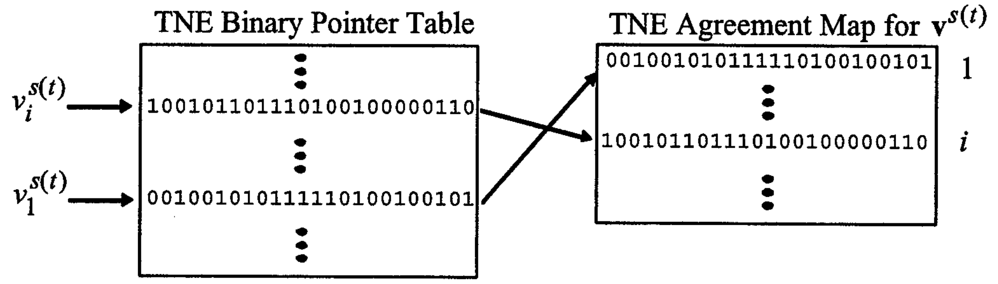


Figure 6.1. *Schematic diagram of how the TNE agreement map for* $\mathbf{v}^{s(t)}$ *is obtained from the TNE binary pointer table.*

Suppose an ATR classification problem has k target classes. Then, it is possible to train a morphological perceptron using a training set $\{(\mathbf{w}^{s,\xi}, c_\xi) : \xi = 1, \ldots, m\}$, where $c_\xi \in 1, \ldots, k$ and $\mathbf{w}^{s,\xi}$ belongs to class $j \in \{1, \ldots, k\}$ if and only if $c_\xi = j$. As shown in [Rit98], training is very fast and the training set will always be correctly classified. After we do this training for each sensor $s = 1, \ldots, M$, the morphological perceptron will assign each test vector obtained from actual observation to a particular target class. For a suite of M sensors, the perceptron classification produces an M x K target classification table, whose column sums will yield the most likely target. More precisely, if

$$k_j = \sum_{i=1}^{m} x_{i_j} > k_l \tag{6.2}$$

for every $l = 1, ..., k,\ l \neq j$, then the target belongs to class j. Here, $x_{i_j} = 1$ if $w_i^{s(t)}$ has been classified by the morphological perceptron as belonging to class j, else $x_{i_j} = 0$. Figure 6.2 illustrates this process.

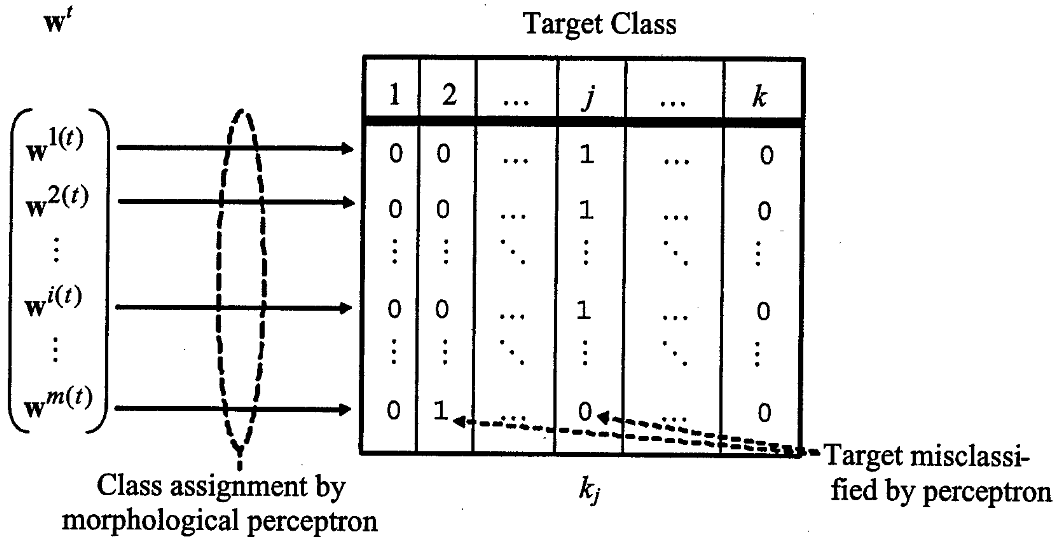

w^t
Target Class
1
2
j
k
w^{1(t)}
w^{2(t)}
w^{i(t)}
w^{m(t)}
Target misclassified by perceptron
Class assignment by morphological perceptron
k_j

Figure 6.2. AM assignment to a target class by the morphological perceptron technique.

6.2. Dendritic Computation for Processing Significant AM Information

The schematic diagram of the processing sequence shown in Figure 6.3 illustrates the simple scheme of summing across components on an AM and/or across (over time) multiple AMs. Obviously, summing across components may help identify a possible target, but often provides only partial and sometimes conflicting information as it does not account for contiguous values of ones. For example, given a vector $\mathbf{x} = (x_1, x_2, ..., x_{10})$ obtained fom a row of a TNE AM, where $x_1 = x_4 = x_8 = x_{10} = 1$ and $x_i = 0$ for $i \neq 1,4,8,10$, then $\sum_{i=1}^{10} x_i = 4$. However, it may be statistically more significant to know if the value 4 thus obtained represents contiguous ones.

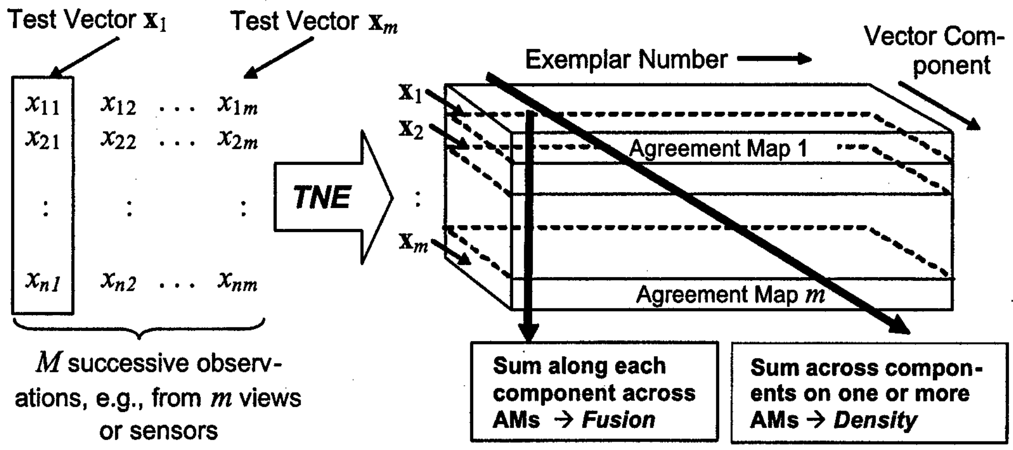

Test Vector $\mathbf{x}_1$
Test Vector $\mathbf{x}_m$
x_{11} x_{12} ... x_{1m}
x_{21} x_{22} ... x_{2m}
x_{n1} x_{n2} ... x_{nm}
M successive observations, e.g., from *m* views or sensors
TNE
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_m$
Exemplar Number
Vector Component
Agreement Map 1
Agreement Map *m*
Sum along each component across AMs → *Fusion*
Sum across components on one or more AMs → *Density*

Figure 6.3: *Schematic diagram of AM processing using component summation: We are free to operation on the 3D binary AM (a) along exemplar components in single or multiple agreement maps, or (b) by component(s) across multiple agreement maps. Thus, MNNs can access the agreement map space in multiple ways, to increase versatility and robustness, while at the same time (i) computing target density per test vector by summing across components, and (ii) fusing multiple inputs expressed as multiple agreement maps, by summing along each component.*

To obtain this type of information from a collection of AMs, we can employ a simple feedforward morphological neural network that needs no training. For the above example with

$$\mathbf{x} = (1, 0, 0, 1, 0, 0, 0, 1, 0, 1)$$

the output of our feedforward MNN would be

$$\mathbf{y} = (1, 0, 0, 1, 0, 0, 0, 1, 0, 1).$$

However, if

$$\mathbf{x} = (0, 1, 1, 1, 1, 0, 0, 0, 1, 0)$$

the output of the MNN would then be

$$\mathbf{y} = (0, 4, 1, 1, 1, 0, 0, 0, 1, 0).$$

In the latter case, we obtain two types of information:

1. $\Sigma_{i=1}^{10} x_i = 5$, which tells us that there are five positive responses, and
2. $\vee_{i=1}^{10} x_i = 4$, which indicates that among the five positive responses, four are contiguous.

This type of feedforward network with both excitatory and inhibitory neurons has been described in [Rit02]. To provide a partial example and illuminate the concept underlying the network, we consider a simple case, where $\mathbf{x} = (x_1, x_2, x_3)$. Figure 6.4 schematically illustrates this network and provides all the axonal branches terminating on output neuron y_1. The inhibitory neurons are denoted by z_i. The activation function for the inhibitory neurons is defined by

$$f(z) = \begin{cases} 0 & \text{if } z \geq 0 \\ \infty & \text{if } z < 0 \end{cases}. \tag{6.3}$$

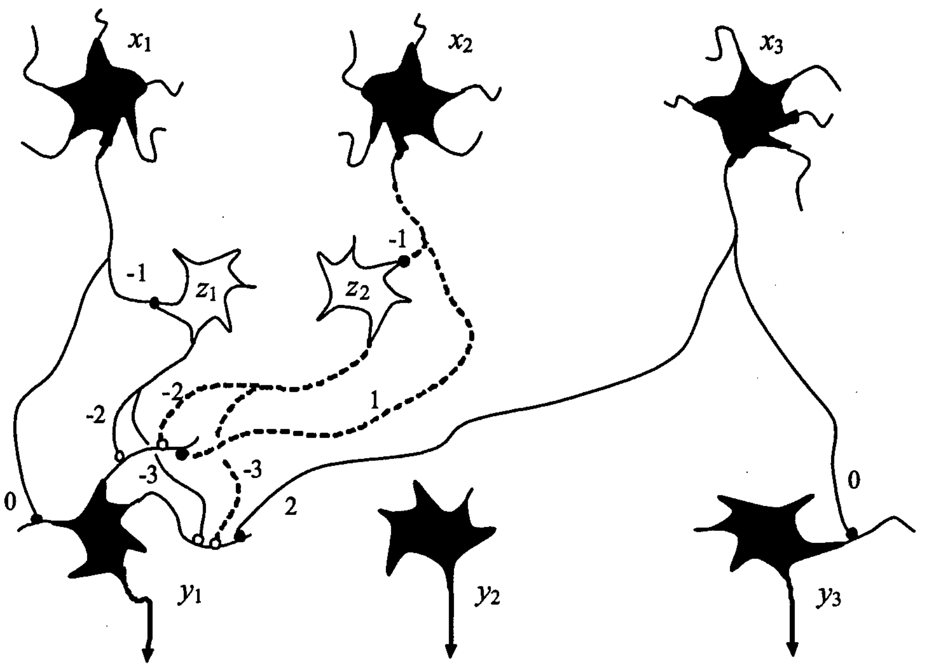


Figure 6.4. *Simple example of AM processing network for three-input test vector* $\mathbf{x} = (x_1, x_2, x_3)$*, showing the axonal branches and dendritic tree for output neuron* y_1 *only. The neurons labeled* z_1 *and* z_2 *are inhibitory neurons. The dashed lines are intended only to distinguish axons associated with* x_2 *and* z_2 *from those associated with other neurons.*

The input neurons have no activation function, but simply send their values down their axons and subordinate axonal branches. The output neurons also have no activation functions as we want to read the total computed value. The computation of the j^{th} output neuron is given by the following formula:

$$y_j = \bigvee_{k=1}^{D_j} \tau_k^j(\mathbf{x}), \tag{6.4}$$

where $\tau_k^j(\mathbf{x})$ denotes the computed value of the k^{th} dendrite of output neuron j and D_j denotes the total number of dendrites of output neuron j. The value of $\tau_k^j(\mathbf{x})$ is given by

$$\tau_k^j(\mathbf{x}) = \bigwedge_{i \in I(k)} \left(x_i + w_{ijk}\right) \bigwedge_{i \in J(k)} -\left(z_i + v_{ijk}\right), \tag{6.5}$$

where x_i denotes the value of the i^{th} input neuron,
z_i denotes the value of the i^{th} inhibitory neuron,
$I(k)$ denotes the set of input neurons having terminal branches on the k^{th} dendrite of output neuron j,
$J(k)$ denotes the set of inhibitory neurons having terminal branches on the k^{th} dendrite of output neuron j,

w_{ijk} denotes the axonal branch weight of the i^{th} input neuron terminating on the k^{th} dendrite of output neuron j, and

v_{ijk} denotes the axonal branch weight of the i^{th} inhibitory neuron terminating on the k^{th} dendrite of output neuron j.

Thus, to compute the value of $\mathbf{x} = (x_1, x_2, x_3) = (1, 1, 0)$, we have

$$y_1 = \bigvee_{k=1}^{3} \tau_k^1(\mathbf{x}) = \tau_1^1(\mathbf{x}) \vee \tau_2^1(\mathbf{x}) \vee \tau_3^1(\mathbf{x})$$

$$= (1+0) \vee [(1+1) \wedge -(0-2) \wedge -(0-2)] \vee [(0+2) \wedge -(0-3) \wedge -(0-3)]$$
$$= 2$$

Similarly for $\mathbf{x} = (1, 1, 1)$ we obtain $y_1 = 3$, while for $\mathbf{x} = (1, 0, x_3)$ and $\mathbf{x} = (0, x_2, x_3)$ we respectively obtain $y_1 = 1$ and $y_1 = 0$, where x_2 and x_3 can have the value 0 or 1.

6.3. Implementational Advantages of AM Processing with MNNs

Due to the potential size of the agreement map, processing of the AM can be potentially burdensome. We have elsewhere shown that it is possible to process the TNE agreement map with Boolean operations implemented in a parallel or reconfigurable processor [Key99] or network of processors. However, MNNs present the advantage of a fundamental increase in processing efficiency for performing contiguity detection *simultaneously with target classification in* **O**(log n) time. This concurrency cannot be realized in traditional vector processing techniques without prohibitive replication of hardware. Analysis and several examples in support of these claims follow.

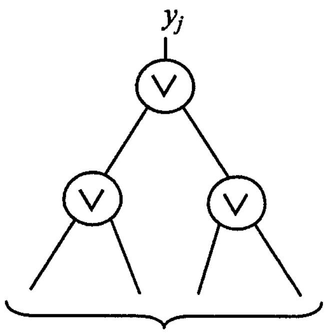
As before, assume that we have n vector components, m sensors, and k targets possible. The agreement map shown schematically in Figure 6.5 thus has kmn elements, each of which is shown I that particular example as one bit. Thus, any operation that processes all locations in the agreement map at least once (as exemplified in Figure 3 by the bold arrows indicating summation) will incur work W = **O**(kmn). Since k can be quite large (e.g., 10^3 to 10^6 in practice), and m is typically much smaller (e.g., 1 to 100), the efficiency of TNE's AM processing algorithm(s) depend(s) on making n as small as possible without losing source resolution by introducing excessive sampling error [Key99]. Assuming that $16 \leq n \leq 256$ is usual, we have $1.6 \times 10^4 \leq kmn \leq 2.56 \times 10^{10}$, a range of over two orders of magnitude.

However, as discussed in Section 5, MNNs can process input in one pass of the network. However, in the example given in Section 6.2, Equation (6.5) requires n^2 partial minima to be computed. Fortunately, this can be computed on a tree-structured architecture, leading to **O**(log n^2) = **O**(log n) time complexity and **O**(n) space complexity, since the processing elements from stage i can be recycled in the computation of stage (i + 1) if double buffering is employed. This concept is illustrated notionally in Figure 6.5.

In particular, the number of processing operations required to compute Equation (6.4), shown in the upper part of Figure 5.5a for $n = 4$, and Equation (6.5), is given for each y_j by

$$W(k) = 3 + |I(k) - 1| + |J(k) - 1| \text{ comparison operations} + |I(k)| + |J(k)| \text{ additions.}$$

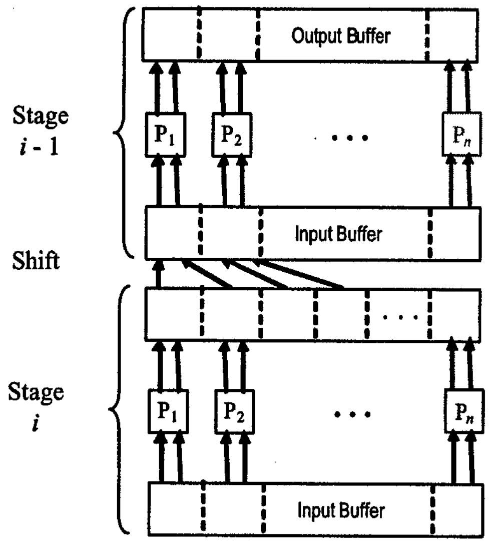
This implies that a tree-structured architecture could compute Equation (6.4) in **O**(log$|I(k)|$ + log$|J(k)|$) addition operations, since comparison is typically implemented with subtraction, which is implemented in terms of addition.



Each leaf is an instance of the term $\tau_k^j(\mathbf{x})$, given in Equation (6.5), as

$$\boxed{\tau_k^j(\mathbf{x}) = \bigwedge_{i \in I(k)} \left(x_i + w_{ijk}\right) \bigwedge_{i \in J(k)} \left(z_i + v_{ijk}\right)}$$

(a)



(b)

Figure 6.5: *Computation of MNN result with a parallel tree-structured architecture: (a) diagram of partial minima and global maximum computation for n = 8, (b) double buffering used to recycle processors for multi-stage hierarchical pipelining of MNN computation. Observe that, in a tree-structured architecture, processor P_n could be used in Stage i but not in Stage i – 1.*

6.4. References

[Key99] Key, G., M.S. Schmalz, and F.M. Caimi. "Performance analysis of Tabular Nearest Neighbor Encoding for joint image compression and ATR", *Proceedings SPIE* **3814**:115-142 (1999).

[Rit98] Ritter, G.X., P. Sussner, and J.L. Diaz de Leon. "Morphological associative memories", *IEEE Transactions on Neural Networks* **9**(2):281-293 (1998).

[Rit02] Ritter, G.X. and G. Urcid. "Lattice algebra approach to single neuron computation", in *IEEE Transactions on Neural Networks* (Nov 2002).

7.0 Conclusions and Future Work

Research at FTI and UF on Phase-I of the GASP project focused on establishing the theoretical and practical basis for MNN-based processing of multiple TNE agreement maps. Emerging technologies such as Morphological Neural Nets and Dendritic Computing with MNNs were integrated into the research results. All tasks in the SOW were addressed, as shown in Section 1. In this section, we summarize our Phase-I findings, then discuss future work for a possible Phase-II research and development effort.

7.1. Research Findings

The combination of disparate classifier outputs is fraught with difficulty, primarily due to differing sensor parameters (e.g., a spectral detection process versus a time-of-flight based detection) and data quality (e.g., high computational error corrupting one classifier output, while another classifier is sensitive to image detector noise propagated through one or more of the aforementioned postprocessing computations). Various techniques such as linear or nonlinear combination, Bayesian or unsupervised classification, and supervised learning techniques such as neural networks have been proposed for refining the output of individual classifiers or combining such outputs to produce a more accurate classification.

Accordingly, we have investigated Bayesian classifiers (Section 3), Frontier Technology's TNE pattern recognition paradigm (Section 4), and UF's Morphological Neural Networks (MNNs) pattern recognition paradigm (Section 5) for both classifier and post-classifier processing, as described in Section 2. Furthermore, we have investigated and formulated approaches for processing the TNE agreement maps using MNNs (Section 6). The latter techniques represent potentially important new developments in pattern recognition, which promise greater accuracy and information storage capacity, as well as improved refinement of classified sensor output.

We have found that nonadaptive Bayesian classifiers, especially in a fixed, rule-based pattern recognition paradigm, lack sufficient flexibility to track input changes (e.g., nonstationarities) in a manner that preserves or increases Pd and minimizes or decreases Rfa. Several additional problems with classical Bayesian pattern recognition theory include the unrealistic requirement that all probabilities be known and accurately computed prior to classification. This stipulation can only be satisfied with full model-based coverage of the input pattern space, which has been frequently shown to be infeasible in ATR field practice. It is well known that model-based ATR often leads to brittle classifiers that incompletely cover input space because they are designed to account for known instances of input patterns. However, the interesting phenomena in ATR field practice are usually those of which one has little or no foreknowledge. It is these unforeseen effects that typically render a target classifier brittle or failed in practice.

An alternative to pure Bayesian classifiers is FTI's TNE pattern recognition paradigm, which is grounded in Bayesian pattern recognition theory at a high level, but allows for uncertainty in the pattern space, noise or partial information in test vector input, and can combine classifier results in a data fusion paradigm. Another alternative is neural network based pattern classification, which can be rendered adaptive via a learning algorithm. Dr. Ritter has developed Morphological Neural Nets (MNNs) as highly accurate, efficient classifiers that are designed to replace classical neural networks in a wide variety of signal and image processing applications. MNNs have the advantages of theoretical maximum information storage capacity, fast and accurate convergence (usually in one pass of the net), and short training times. Additionally, theory that predicts the performance of MNNs under partial information or noise has been published in the literature and summarized in this report.

In this Phase-I research effort, we examined how MNNs could be applied to processing of the TNE agreement map to further increase classification accuracy. If one views TNE's target database, distance function, and thresholding parameter(s) as *a priori* knowledge, then one can think of the columns of TNE's agreement map as the partial classifier results. We have shown how MNNs could be applied to these columnar partitions to (a) render partial classification more robust, (b) combine results from adjacent columns to increase the likelihood of accurate target classification in the classify-before-detect ATR paradigm, and (c) direct further statistical processing or optimization of MNN/TNE results.

In Section 4, Section 5, and Appendix A, we discuss two advanced technical issues, namely (1) the use of array processors to compute the TNE or MNN algorithms, and (2) the use of pointwise versus componentwise performance measures of Pd and Rfa to trigger MNN processing of the agreement map. Since TNE and MNNs are based on matrix operations (e.g., vector correlation and pointwise bitmap operations in TNE, with nonlinear vector-matrix products in MNNs), it is possible to parallelize these operations for computation on embedded SIMD array processors or reconfigurable computers such as field programmable gate arrays (FPGAs). We showed that, for a target database comprised of M exemplars, classification of N samples of K pixels each could be computed in W = **O**(KMN) work. In practice, since the bits that comprise the agreement map can be operated upon in parallel fashion (e.g., Boolean vector processing), it is possible for W to have a small proportionality constant. We also showed that MNNs could compute in **O**(N(logK + logM)) time, given sufficient parallelism and I/O bandwidth.

7.2. Future Work

Because of our concerns pertaining to classifier performance in the presence of nonstationary input, we propose follow-on research (e.g., Phase-2 of the current STTR) to develop adaptive pattern recognition capabilities for TNE that would be directed by classification performance results. We propose to achieve this objective via the following innovations:

1. ***MNN-directed TNE target database optimization*** – It is well known that template-based pattern recognition systems perform more efficiently given a smaller number of target templates. Consider the problem of detecting multiple rotated versions of the same target. Although it may be difficult to analytically determine the optimal rotation angle quantization level, this can be done empirically by (a) merging similar target templates, then (b) constructing a Receiver Operating Characteristic (ROC) curve from Pd and Rfa analysis of TNE classifier output obtained from ATR imagery. When Pd and Rfa begin to underperform with respect to specifications, then it can be argued that the optimal quantization level has been exceeded. However, this tells us nothing about the underlying merging process inherent in target template combination, and its effect on TNE classifier performance. Neither does this assumption consider cosine or tangent projection laws that cause adjacent views in out-of-plane rotations near 45 degrees to appear significantly different visually, as opposed to out-of-plane rotations near zero degrees, which appear almost identical. Apparently, a nonlinear rotation angle quantization scheme would be needed to compensate for such projection effects.
   Proposed Work. In Phase-II, we propose to investigate the concept of *adaptive quantization* using MNNs to cluster rotated targets according to the expected viewpoint. This approach, which we call *viewer-adaptive training set optimization*, would allow the inter-view spacing (e.g., angular quantization step size) to be varied as a function of out-of-plane rotation angle as well as performance variables such as Pd or Rfa. Thus, we would more closely approach optimization of the target database from a physical (i.e., viewer-dependent) or mathematical (e.g., ATR filter sensitivity) perspective. Additionally, we propose to explore the effect of noise and contrast degradation (e.g., effects of intensified imaging in night viewing applications) on angular quantization levels, again from a performance perspective grounded in Pd and Rfa based metrics.

2. ***Performance directed target database and AM processing optimization*** – As noted in Item 1), above, when Pd and Rfa no longer meet performance specifications, one should be able to employ automatic re-optimization of the TNE template database, as well as apply MNN-

assisted TNE agreement map processing, as investigated in Phase-1. For example, if a small angular quantization interval is selected for rotated targets, this might be increased for night viewing, due to the effects of image intensifier noise on (a) image resolution and (b) resultant TNE correlation performance between a test vector and each pattern in the template database. We note that there are many other parameters, in addition to angular resolution, that can serve as bases for determining whether or not to cluster targets in the TNE database. These should also be investigated, as described below.

Proposed Work. In Phase-II, we propose to investigate (a) parameters that could be used to direct template clustering in the TNE database, as well as (b) theory and algorithms for relating TNE database structure to TNE performance in terms of Pd and Rfa. Example parameters include but are not limited to camera noise, measures of input nonstationarity, target/background contrast and target rotation/scaling effects. Although there has been some investigation of these effects for TNE-based compression, additional research is required to transfer these technology developments and insights to the less constrained application domain of target recognition. Additionally, we propose to investigate how MNNs could be used to cluster TNE templates based on well-established similarity or content metrics published in the literature.

3. *MNN-assisted TNE classifier refinement* – Given results produced in the Phase-1 investigation, it is readily apparent that MNNs have considerable present and future utility in TNE agreement map processing. At the present time, MNNs can determine saliency of partial classification results (e.g., columns in the TNE agreement map) or can be configured to operate on the agreement map to combine columns in a data fusion paradigm. We would also note that it is possible to process the agreement map using statistical procedures such as Analysis of Variance (ANOVA) to determine diagonal orientations of detections within columns of the agreement map, which can indicate phase shifts among similar targets or target types. We thus propose the following work items for a possible Phase-II study.

Proposed Work. In Phase-II, we propose to extend the research performed in Phase-I, wherein we investigated the feasibility of applying MNNs to process the TNE agreement map. In particular, we propose to investigate techniques for interrogating the agreement map while employing other than rowwise or columnwise partitions. In Phase-I, we found that interrogation of AM rows, or spatially clustered groups of rows, supported preliminary selection of possible targets that were classified and detected within those spatial partitions. This information could be used to downselect additional target templates for multi-pass recognition, or to select strategies for measuring or analyzing Pd or Rfa within TNE sampling partitions. In the latter case, this would partly support sub-block detection (e.g., for small features or targets) under Pd and Rfa constraints or metrics that are customized for the expected target types. Additionally, in Phase-1, we found that columnwise processing with MNNs could implement data fusion through combination of partial classification results specific to similar target instances or types.

In Phase-II, we propose to extend these research results to search for other types of detection patterns within an agreement map. This could be done by a spatially based ANOVA to detect, for example, diagonal patterns indicative of phase shifts. As an illustration of this concept, consider the situation where the cannon or turret of a tank is detected in a digital image. However, this turret might be in a slightly different position in different tank templates, due to lack of template co-registration as well as scale or rotation differences. Rather than scanning the AM row-wise, it would be useful to detect whether there were phase shifts within a group of content-related templates (e.g., similar tank types with similar out-of-plane angular rotations). In practice, this approach would be useful for refining the classification result to achieve more accurate target identification.

The result of the proposed future work would be an *adaptive* and more *highly refined* target or *pattern classification paradigm*. In this envisioned GASP system, which we outlined in the Phase-1 proposal, ATR imagery would be processed by TNE's sampling module and applied to the TNE engine to yield an agreement map. The TNE engine would employ a target template database that would be optimized as described previously, to adapt to changing input conditions according to Pd and Rfa results. MNNs would process the TNE agreement map, as described in item 3), above, and these MNNs would also be constrained by Pd and Rfa. If time and resources permit, it could be useful to investigate MNN-directed adaptation of TNE for target classification in variable-noise scenarios typical of intensified cameras (e.g., night vision applications).

The proposed research would produce a powerful fusion of neural nets and TNE to yield adaptive classification compensated for physical effects such as noise that manifest in the pattern space. This represents a new direction in pattern recognition with potentially high payoffs in a wide variety of military and domestic applications.

Appendix A: Advanced Technology Summary

Several technology areas-of-need have been highlighted in this Phase-1 study. The two most prominent areas are (1) embedded parallel or reconfigurable processor technology to support TNE computation (discussed in Section B.1) and (2) the use of pointwise versus componentwise measures of Pd or Rfa to trigger MNN-based codebook optimization or MNN-directed processing of the TNE agreement map (discussed in Section B.2).

B.1. Embedded Parallel Computation of TNE

This section contains a high-level design and summary analysis for an ATR architecture based on (a) the tabular nearest-neighbor (TNE) pattern recognition paradigm, (b) one or more reconfigurable processors, and (c) a scalable multi-port I/O system. In particular, the proposed architecture is based on a common bus that is designed to facilitate switching of binary input (two operands) to selected reconfigurable computer (RC) units. The processing units are designed to be reconfigurable to support different operations over the agreement map (AM), a data structure key to the implementation of TNE. Since the key challenge of implementing TNE on RCs is the I/O overhead associated with retrieval of the agreement map rows, followed by column operations over the agreement map, we first emphasize I/O cost. This approach is germane due to low computational cost – in practice, addition and Boolean logic operations only are required. Performance estimates presented herein are highly conservative, and would be increased considerably by more advanced parallel or reconfigurable processing technology that will likely be available at the time of possible prototype construction.

B.1.1. Introduction to Parallel Reconfigurable Processor Design Issues: Frontier Technology's TNE pattern recognition paradigm requires retrieval of bit vectors from memory containing preprocessed templates (PTM). Each bit vector comprises a row of an intermediate data structure called an agreement map, which changes with each position of a source image sampling window. In this context, a system designer's first concern should be the I/O operations required to move information in and out of the agreement map. A second design issue, which is no less important, is how to efficiently retrieve AM data from the PTM and transfer it into an RC processor, such that the processor can operate on the AM data efficiently. Additional design issues include, but are not limited to, minimization of the degree of parallelism, degree of memory port replication, and complexity of the RC processing circuitry.

B.1.1.1. Assumption. Given (a) K pixels per sampling window or block applied at selected domain points of an N-pixel source image, (b) a maximum of K pixel shifts per block, and (c) K bit vectors retrieved from memory comprised of L bits per row vector, the following minimum I/O work is required for AM access, during processing of a source image sequence comprised of F frames per second (fps) with R replicates per frame:

$$W_{I/O} \leq KNLFR \text{ bits/sec (bps).} \qquad (B.1)$$

Under the prespecified constraints, $W_{I/O}$ describes the approximate number of I/O operations required to move the AM data *only*. Note that $W_{I/O}$ does not include I/O overhead required for RC reconfiguration or for moving source data or processed output (results) into buffer memory.

B.1.1.2. Example. If K = 256 pixels, N = 1M pixel, L = 15,000 templates, F = 1 fps, and R = 1, then KNLFT = 4.02 terabits per second (Tb/s). At video rates (F = 30fps), this product increases to 120 Tb/s. This is clearly a problematic data rate for architectures based on relatively slow I/O bus technology, such as field-programmable gate arrays (FPGAs).

B.1.1.3. Observation. The work requirement shown in Equation (B.1) can be ameliorated by subsampling each (a) source block (sampling window) at a fraction f_K of K pixel shifts, (b) image at a fraction f_N of N possible source block positions, (c) AM at a fraction f_L of its L columns, and (d) image sequence at fractions f_F of frame rate F and f_R of R frame replicates. The work requirement can thus be expressed as

$$W_{I/O} \geq (f_K \cdot K)(f_N \cdot N)(f_L \cdot L)(f_F \cdot F)(f_R \cdot R) \text{ bits/sec (bps),} \qquad (B.2)$$

which can be rendered tractable to current memory, bus, and RC technology if the factor

$$f = f_K \cdot f_N \cdot f_L \cdot f_F \cdot f_R$$

is kept sufficiently small.

B.1.1.4. Example. Given the values K = 256 pixels, N = 1M pixels, L = 15,000 templates, F = 1 fps, and R = 1, let $f_N = 0.1$ (e.g., 10 percent of source points underlie a pixel value deemed to be part of an interesting candidate target). Further let $f_K = 0.5$ (i.e., half of the pixels in the sampling window are randomly selected), and let $f_L = 0.1$ (i.e., there exists a tenfold reduction in AM size, for example via a hierarchical access scheme). Given $f_F, f_R = 1$, $W_{I/O}$ is computed from Equation (B.2) as:

$$W_{I/O} \geq (0.5 \cdot 256)(0.1 \cdot 1M)(0.1 \cdot 15{,}000) \text{ bps} = 20.13 \text{ Gbps}.$$

For example, this estimate of I/O might be feasible using high-bandwidth multiport I/O and parallel processing technology, as discussed below.

B.1.1.5. Observation. Observation B.1.1.3 implies that several design strategies could be concurrently employed to mitigate the I/O cost in the presence of relatively low FPGA clock speed supported by current technology (e.g., 200 MHz to 500MHz). A common remedy for this situation is *I/O parallelism*. In a practical parallel processing scenario, I/O cost could be shared among P processors, each of which could have M I/O ports. In that case, the maximum reduction in I/O cost would be expressed as the product PM.

Another technique that appears to be applicable to TNE is *hierarchical partitioning* of the AM domain, in particular, the column coordinate. For example, one could group a nonzero fraction f_L of L templates specific to a given target T to yield $L_H = f_L \cdot L$ high-level templates. Each of these high-level templates could be associated with a lower-level partition of template space, of mean size L/L_H templates. The best-match score at a high level would point to the most likely partition to be searched at a lower level. This strategy, which FTI personnel have indicated is feasible in practice, would reduce $W_{I/O}$ by an average amount f_L.

A third method that may help reduce AM I/O cost is *pipelining* of AM I/O, which could be achieved by dividing the AM into n-column partitions of Kn pixels each, where $n << Q$. This could facilitate processing of the AM on a network of P RCs, each having storage capacity of Kn bits. Following a brief, high-level description of the proposed architecture, each of the preceding methods is subjected to summary analysis.

B.1.2. Proposed Prototype Architecture. Let P processors denoted by $P_1, P_2, ..., P_P$ having local memories $M_1, M_2, ..., M_P$ be connected by a common bus comprised of two m-bit input buses B_1, B_2 and one 2m-bit output bus B_3, as shown in Figure B.1.

B.1.2.1. Data Bus and Control. Let each processor have input (output) switched from (to) the common bus by m-pole (2m-pole) single-throw normally-open switches. For example, if P_i accepts input from B_1 or B_2, then input switches $S_{i,1}$ or $S_{i,2}$ would be closed, allowing data to flow to P_i. (The output switch for the i-th processor is denoted by $S_{i,3}$.) Each of the three banks of switches could be controlled by a separate control bus of maximum width $\lceil \log P \rceil$ bits, denoted by C_1, C_2, or C_3.

Alternatively, the aforementioned switches could be controlled by a single bus of width $\lceil \log 3P \rceil$ bits. Although such multiplexing might save bus space, the presence of demultiplexing hardware is implied, which could increase circuit complexity, power consumption, and (possibly) I/O time.

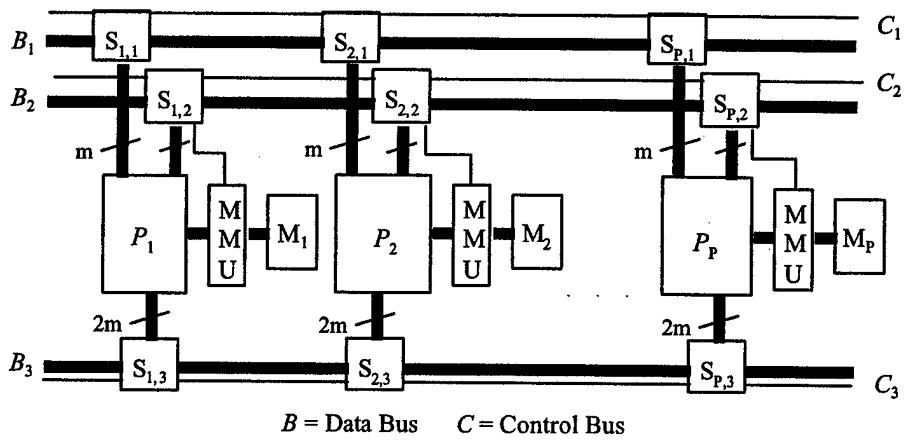


Figure B.1. *Schematic diagram of common-bus architecture for RC based processing of TNE ATR operations, where P_i denotes the i-th processor, $S_{i,1}$, $S_{i,2}$ ($S_{i,3}$) denote the input (output) switches specific to P_i, MMU denotes a memory management unit, and M_i denotes P_i's local memory.*

B.1.2.2. Features and Advantages: The common bus would allow a partition of data to be broadcast to multiple processors, e.g., a K-pixel sampling block in TNE, or template/parameters required by non-TNE paradigms such as Boolean template matching or MISD-parallel processing of multiple ATR filters applied to a given input. The switches would allow different processors to receive specific data partitions or (in the case of TNE) MMU instructions that could facilitate retrieval of AM data from local memory. The latter operation could be implemented via bypassing the FPGA with a separate control bus derived from B_1 or B_2, as shown in Figure B.1.

B.1.2.3. TNE Processing Elements: In practice, TNE produces in each column of the agreement map a vector of bits, each of which indicate whether or not a vector component matches the corresponding value in a template at a particular pixel position and greylevel. The summation of the j-th column of the AM indicates at what level the sampling block matches the j-th template. This can be thought of as somewhat similar to the Hamming distance that might be computed by matching a source block to the j-th template, under a prespecified thresholding criterion. Thus, reconfigurable processing elements (PEs) for TNE-based ATR could in principle be relatively simple, since each processor could initially compute the sum of each column in a partition of AM columns assigned to that processor. This would yield a measure similar to the Hamming distance between the current sampling block and each of the templates that correspond to the given partition of the AM column.

Given the Boolean set $\mathbf{B} = \{0,1\}$, for purposes of architectural development we adopt as our example application computing the sum of each column $\mathbf{a}^j \in \mathbf{B}^K$ of an agreement map denoted by $\mathbf{a} \in \mathbf{B}^{K\times Q}$. The Hamming-like distance that corresponds to $\mathbf{a}^j$ can thus be expressed as:

$$D_H(\mathbf{a}^j) = \sum \bar{\mathbf{a}}^j \ , \qquad \text{(B.3)}$$

where the overbar denotes complement. Assuming that K = 2^m, the equivalent image algebra expression

$$\sum ?_{=1}(\bar{\mathbf{a}}^j) \equiv \sum \bar{\mathbf{a}}^j$$

implies that a *modulo*-2^m counter with inverted input could be employed to compute D_H.

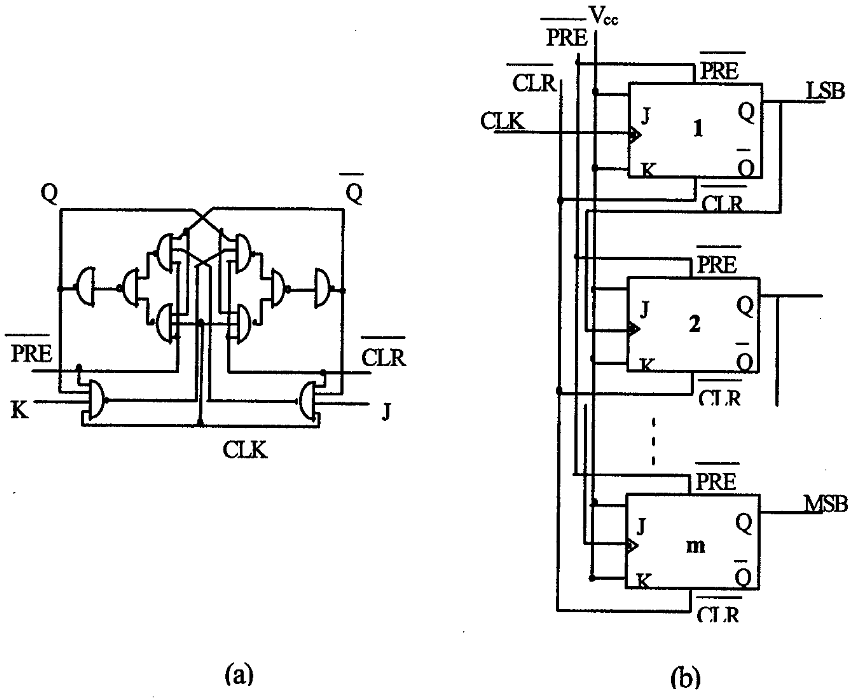


Figure B.2. Schematic diagram of (a) J-K flip-flop constructed of 10 nand gates, and (b) a modulo-2^m counter constructed from m J-K flip-flops.

B.1.2.4. Example. In a prototype architecture, it is possible that such counter circuitry could be contained within a functional block of each processor P_i. Using existing tools, it would be possible to specify such functionality in VHDL code, which could be implemented in reconfigurable logic with *nand* or *nor* gates, for example, as shown schematically in the m-bit counter of Figure B.2. Here, 10m (8m) *nand* (*nand* + *nor*) gates are required for a *modulo*-2^m synchronous counter. The input would be inverted to implement the Boolean complement operation shown in the preceding equation.

B.1.2.5. Space Complexity. Assume that a given PE can be configured without incurring performance penalties during execution due to I/O pathlength dictated by the VHDL-to-hardware place-and-route algorithm. If N_g *nand* gates are required per counter, then one could additionally assume an overhead f_g as a fraction of total gates, for I/O handling (buffering, etc.), thereby yielding a total of $N_g(1+f_g)$ gates per *modulo*-2^m counter.

For purposes of simplicity, assume that each configurable logic block (CLB) in an FPGA has an integral number of counters only, due to limitations on I/O into or out of each CLB. Letting each FPGA have N_b CLBs with N_{bg} gates per block, the maximum number of counters per FPGA fully populated with counter circuits would be estimated as

$$N_c \leq N_b \cdot \left\lfloor \frac{N_{bg}}{N_g(1+f_g)} \right\rfloor . \tag{B.4}$$

Solving the preceding equation for N_g yields the following conservative bound on the number of gates per counter:

$$N_g \geq \left\lceil \frac{N_b}{N_c} \right\rceil \cdot \frac{N_{bg}}{(1+f_g)} , \tag{B.5}$$

where the ceiling function reflects the previous assumption pertaining to the number of counters per CLB.

B.1.2.6. Example. If $f_g = 0.2$, K = 256, each counter requires $N_g = 8 \cdot \log K$ gates, and an FPGA is partitioned into $N_b = 1024$ (32x32-element array of) functional blocks of $N_{bg} = 100$ gates each, then

$$N_c \leq 1024 \cdot \left\lfloor \frac{100}{64(1+0.2)} \right\rfloor = 1024 \text{ counters}$$

could be implemented per FPGA, neglecting other on-chip requirements.

B.1.2.7. Observation. Each FPGA functional block may require that additional gates be allocated to non-computational work such as buffering data routed to and from local memory. Until the counters are coded in VHDL, and a detailed schematic is developed, it is not possible to know how many additional gates would be required.

B.1.3. Preliminary Analysis. We begin with an efficiency analysis, then concentrate on I/O and computational cost analysis specific to the implementation of TNE/ATR on the architecture of Figure B.1.

B.1.3.1. High-Level Efficiency Analysis. Our Phase-1 research indicates that the key constraint on TNE's implementational success is the efficient retrieval of agreement map data from the local memory of each processor. Proceeding from the general to the specific in this analysis of efficiency and resource constraints and given an execution time per block in the proposed RC-based architecture (operating in non-pipelined mode) of Δt_B, we note (from Equation B.1) that the total execution time per frame is given by

$$\Delta t_{tot} = NFR \cdot \Delta t_B ,$$

where the source frame size is denoted by N, frame rate by F, and replication factor by R.

Letting F,R = 1 for purposes of simplicity, the total computational bandwidth can be expressed as

$$BW_{tot} \leq \frac{1}{N \cdot ?t_B} . \qquad \text{(B.6)}$$

We next consider the role of I/O in the formulation of Δt_B.

B.1.3.1.1. I/O Constraints. Assume that the L columns and K^2 rows of the template database are partitioned P ways, such that each local memory M_i contains $n = \lceil L/P \rceil$ columns of the agreement map. This implies that the size of M_i is given by

$$|M_i| \geq K^2 \cdot \lceil L/P \rceil \text{ bits} . \qquad \text{(B.7)}$$

B.1.3.1.1.1. Example. If K = 256, L = 16K bits, and $P \leq 16$, then $|M_i| \geq 256^2 \cdot 1\text{K bits}$ = 8M bytes. If P = 4 (a more reasonable value), then $|M_i| \geq 256^2 \cdot 4\text{K bits}$ = 32M bytes, which is achievable with current technology.

Assume that the dominant component of Δt_B is the I/O time required to retrieve AM data from local memory. (This assertion is supported by the analysis of Section B.1.3.2.) Letting Δt_{mem} denote the memory cycle time and N_{mp} the number of memory read ports, we have the following I/O-dominant bound per block:

$$\Delta t_B \geq \frac{Kn \cdot ?t_{mem}}{N_{mp}} ,$$

which can be inverted to obtain block-specific bandwidth. Via substitution of the preceding equation into Equation (B.6), total bandwidth can be elaborated as

$$BW_{tot} \leq \frac{1}{N \cdot ?t_B} = \frac{N_{mp}}{(f_N \cdot N)(f_K \cdot K)(f_L \cdot n) \cdot ?t_{mem}} , \qquad \text{(B.8)}$$

where the fractions f_N, f_K, and f_L were defined previously. Solving for n, the bound on the number of AM columns stored in local memory is given by

$$n \leq \frac{N_{mp}}{(f_N \cdot N)(f_K \cdot K)(f_L \cdot BW_{tot}) \cdot ?t_{mem}} . \qquad \text{(B.9)}$$

B.1.3.1.1.2. Example. Letting f_N, f_K, f_L = 1 for purposes of conservative analysis, and assuming that N = 1M, K = 256, BW_{tot} = 1 Hz, $N_{mp} \leq 4$, and $\Delta t_{mem} = 10^{-8}$ sec (i.e., 10 ns memory cycle time), we obtain $n \leq 4/(2^{20} \cdot 2^8 \cdot 10^{-8}) = 1.49$ bits, which is prohibitively small.

In contrast, letting f_N = 0.1 (10 percent of the source pixels are interrogated), f_K = 0.5 (half of the block contents are sampled), and $\Delta t_{mem} = 8\text{x}10^{-9}$ sec (8 ns memory cycle time), one obtains n = 37 bits, which is insufficiently large to support feasible parallelism.

However, if L = 16K bits, then this implies that the degree of parallelism (i.e., number of processors) P = 16K/37 = 443 processors, which is prohibitive for current FPGA-based architectures. To obtain $P \leq 16$ at F,R = 1, there must be realized a speedup of 443/16 = 27.6:1 or greater.

B.1.3.1.1.3. Observation: A useful method for increasing this processing efficiency is the hierarchical partitioning of the column coordinate of the agreement map. For example, if f_L =

0.1, then n can be brought to within a factor of 2.8 (approximately 11) of the value required for 16-fold (four-fold) parallelism. Over the duration of the proposed Phase-II project, the remaining factor of 2.8 could possibly be compensated by increases in N_{mp} and Δt_{mem} as memory technology develops.

B.1.3.1.1.4. Remark. In practice, a reduction in £ could be achieved through a two- or three-level scheme, in which similar templates are grouped together via Boolean operations on neighborhoods of each row in the agreement map. This would yield a high-level assembly of composite templates, where each template would likely have reduced resolution with respect to the original templates from which it was derived.

B.1.3.1.1.5. Example. If a target is imaged at 10 degree rotational increments (azimuth and zenith), then grouping of adjacent templates in 30 degree disjoint but adjacent azimuth and zenith angle ranges could yield a nine-fold reduction in L (i.e., £ = 0.11).

We next consider the space requirements of the architecture shown in Figure B.1.

B.1.3.1.2. Space Complexity. There are several issues to consider when analyzing space complexity of the proposed architecture, which include (1) bus size and switching, (2) processor size (number of gates), partitioning scheme (e.g., flat-field versus hierarchical), and bus structure, as well as (3) local memory size. We overview each of these issues, then make recommendations for further analysis.

B.1.3.1.2.1. Bus Complexity. As shown in Figure B.1, three data and control buses are sufficient to compute binary image operations on an RC based processor. In Figure B.1, S_1 and S_2 have width m bits, and S_3 has width 2m bits. The control buses C_1 through C_3 each have width $\lceil \log P \rceil$ bits, for a total bus space complexity of

$$\mathcal{S}(\mathrm{m}, \mathrm{P}) = 4\mathrm{m} + 3 \cdot \lceil \log \mathrm{P} \rceil \text{ bits} .$$

Note also that buses do not have convoluted shapes, and could thus be straightforwardly routed on a multi-layer circuit board.

B.1.3.1.2.2. Example. If source imagery has eight bits per pixel and 16-fold parallelism is employed, then m = 32, P = 16, and $\mathcal{S}$(m,P) = 32 + 3(4) = 44 bits, which is not prohibitive in terms of current board fabrication technology. If P = 8 (P = 4), then $\mathcal{S}$(m,P) = 41 bits (38 bits), which are less stringent than the first case.

B.1.3.1.2.3. Processor Size. A key concern for efficient RC-based processing is whether or not the RC processing chip set has sufficient gates to implement the required intra-processor parallelism. In Equation (B.4) we expressed the number of counters that could be derived from a reconfigurable logic chip of N_g gates. As stated, we expect that approximately one counter per CLB could be obtained. This reduces the issue of intra-processor parallelism to determining the number of available CLBs. Let us make a best-case assumption, namely, that all CLBs on-chip would be available for counter implementation, and that all counters could be run simultaneously (e.g., sufficient I/O redirection to permit efficient transmission of each AM column to each RC processor). The feasibility of this assumption is further explored in Section B.3.3.

We have stated that an AM partition size of n = 1K columns could be accomodated in the worst case, that is, in the absence of partitioning of the AM column coordinate (e.g., £ = 1). This implies degree of parallelism of 1K counters (e.g., a 32x32-element matrix of sufficiently large CLBs on-chip) in order to process all columns concurrently. Fortunately, such parallelism is available with current FPGA technology. The case where £ < 1 would obviously present less of an implementational challenge.

B.1.3.1.2.3. RC Processor Partitioning. The majority of FPGAs or RC chips are configured as a flat array of gates, with high-speed communication buses. The regions of the gate array delimited by the buses are called configurable logic blocks. Each CLB has essentially the same bus access priority and can support the same type of circuits as other CLBs. Cross-block mapping is not necessarily problematic, except for exotic memory applications.

In contrast, there exist several hierarchically partitioned FPGA architectures, notably the Actel Eclipse, with three levels of partitioning, namely, CLB, functional block (FB), and gate. In such devices, a given CLB or FB can be programmed to have different bus access priorities, data transfer speed, and gate interconnections, independent of other partitions. As a result, a variety of complex circuits are possible, for example, using small partitions of each CLB for address incrementation, larger partitions for integer adders, and one or more CLBs for multipliers or division circuits.

Challenges involved in the programming of FPGAs include external I/O, inter-block and inter-chip communication, programming hierarchical FPGAs. The issue of external I/O buses has been discussed previously with respect to the problem of obtaining AM data stored in local memory. Communication between CLBs and FPGA chips is supported in various ways by different manufacturers and on different chips. Since FPGA technology appears to be in its late infancy, it is difficult to predict what capabilities will be offered in future devices. However, for the architecture of Figure B.1, it would be obvious to concentrate on high inter-CLB communication bandwidth (i.e., fast internal buses on-chip), as well as support for fast external I/O. For example, Xilinx is a leader in this field, and supports segmented routing, which takes longer to route cross-chip than within-chip. In contrast, Altera offers more efficient cross-chip routing, but penalizes routing within-chip.

The programming of flat-field FPGAs has fewer associated challenges than programming hierarchically-structured FPGAs, due to more degrees of freedom in the latter partitioning model. For example, a VHLD-to-hardware compiler for hierarchical FPGAs must determine not only the arrangement of circuit components (e.g., links between gates in the 2-D gate array) but also must perform near-optimal place-and-route in the presence of a third dimension (i.e., level of paritioning). As a result, we recommend the use of a flat-field Xilinx FPGA (e.g., Virtex series) for a prototype architecture in a possible Phase-II effort.

B.1.3.1.2.4. Local Memory Size. An additional implementational issue is the size of local memory required to store sufficient information for the agreement map partition assigned to P_i. Equation (B.7) succinctly describes the memory size $|M_i|$ required to implement retrieval of an n-column partition of the agreement map. Additional data to be stored in local memory might include an access frequency table (e.g., a 2-D histogram having domain coordinates that correspond to K greylevels and K pixel positions). This table would record the cumulative frequency of access for a given source pixel intensity at a given position, and could be used by an intelligent MMU to facilitate more efficient memory access via a prefetch buffer. For example, if the buffer stored the 256 most frequent accesses to AM data, then the additional space requirement in M_i would equal 256n bits. For n = 1K, (32Mbits overhead) this is not a significant additional cost.

B.1.3.2. Functional Block Timing Analysis. In order to understand the role that each CLB plays in the determination of complexity and cost of TNE algorithm implementation, we present the following algorithm for processing one column of the agreement map.

B.1.3.2.1. Algorithm. Let a K^2-row by n-column partition of the agreement map be stored in memory Mi of processor P_i, where i = 1..P. To retrieve the AM data specific to a given sampling block, then process the data and send results back to the host unit, the following execution sequence is recommended for the architecture of Figure B.1:

Step 1. *Submit Block Contents to Processor* – Host unit closes all switches on B_2 and sends retireval codes down B_2 to the MMU for M_i. Each retireval code consists of a transformation of each pixel $(\mathbf{x},\mathbf{a}(\mathbf{x}))$, where $\mathbf{x}$ is a domain point (pixel coordinates in sampling block) and $\mathbf{a}(\mathbf{x})$ denotes the pixel value. In practice, $f_K \cdot K$ retrieval codes would be broadcast to all P_i.

Step 2. *Retrieve Salient AM Partition* – Each processor could pass the incoming retrieval codes to the MMU from B_2, or the processor could be circumvented via the small interconnecting bus shown in Figure B.1. The latter alternative is faster, but involves more hardware. The MMU would retrieve the $f_K \cdot Kn$ required bits from M_i, which would be sent to P_i.

Step 3. Compare test vector from sampling block with agreement map. For example, compute the Hamming or Euclidean distance between the test vector and each column of the AM.

Step 4. Output comparison results serially from PEs P_i by (a) closing $S_{i,3}$ and ensuring that all other switches on B_3 are open, then (b) sending the results from P_i down B_3, (c) opening $S_{i,3}$.

B.1.3.2.2. Analysis. The preceding algorithm can be analyzed by decomposing each step into its components, to which is assigned a time delay. The following notation pertains:

Table B.1. *Symbols for timing analysis of CLB based processing of a sampling block.*

Symbol	Description
Δt_{BiSW}	Opening or closing of switches on bus B_i
$\Delta t_{BiXM}(n)$	Transmission of an n-bit result along bus B_i
Δt_{MMPR}	Processing of one MMU retrieval code
$\Delta t_{MI/O}(n)$	Retrieve n bits from local memory M_i
Δt_{PiDS}	Compute distance between test vector and AM column

As noted previously, Δt_{PiDS} could represent the time required to compute the Hamming or Euclidean distance between the test vector and a column of the agreement map. Given the preceding notation, an expression of the total processing time for one sampling block being compared with the AM partition retrieved from M_i and stored on P_i is given by the following mathematical model:

$$\begin{aligned}\Delta t = {} & \Delta t_{B2SW} + \Delta t_{B2XM} && \text{[Step 1]} \\ & + K \cdot (\Delta t_{MMPR} + \Delta t_{MI/O}(n)) && \text{[Step 2]} \\ & + n \cdot \Delta t_{PiDS} && \text{[Step 3]} \\ & + P \cdot (2\Delta t_{B3SW} + n \cdot \Delta t_{B3XM}) \, . && \text{[Step 4]}\end{aligned} \qquad (B.10)$$

B.1.3.2.3. Observation. Possible value ranges for variables in the preceding equation include those listed in Table B.2, below. To determine the I/O cost, inspection of the second line of Equation (B.10) shows that $K \cdot (\Delta t_{MMPR} + \Delta t_{MI/O}(n))$ is the dominant term, since the memory cost $\Delta t_{MI/O}(n)$ for $n = 1$ is multiplied by Kn, assuming linearity of I/O cost.

B.1.3.2.4. Example. Let $\Delta t_{MI/O}(n) = n(10 \text{ ns})$, and $n = \text{n} = 1024$, with $\Delta t_{MMPR} = 10$ ns. In this case, the second line of Equation (B.10) would have the dominant term

$$K(n + 1)(10 \text{ ns}) = 2.624 \text{x} 10^{-3} \text{ sec}$$

per source pixel. Note that this is not a function of the type of processor employed (e.g., DSP, embedded parallel processor, or FPGA), but rather an inherent problem in the movement of large amounts of data across local memory buses. In contrast, if n-fold parallelism is available on each processor (e.g., n counters running concurrently to compute D_H applied to each AM partition's column), then the *computational* cost represented by line 3 of Equation (B.10) would be given by

$$1.02 \text{ x } 10^{-4} \text{ sec} < n \cdot \Delta t_{P/DS} < 2.05 \text{ x } 10^{-5} \text{ sec},$$

which is 1/128 to 1/26 of the I/O cost estimated above. Thus, in a realistic model of computation the I/O cost predominates, which validates the discussion of Section B.1.2.

Table B.2. *Example value ranges for analysis of CLB based TNE processing.*

Symbol	Conservative Estimates of Model Parameter Values
$\Delta t_{B/SW}$	1 ns to 10 ns
$\Delta t_{B/XM}(n)$	1 ns/bit to 10 ns/bit (100 Mbps to 1 Gbps)
Δt_{MMPR}	5 ns to 15 ns (fast cache code-to-address translation)
$\Delta t_{MI/O}(n)$	10 ns to 50 ns per access (primary memory assumed)
$\Delta t_{P/DS}$	20 μs to 100 μs (10MHz to 50MHz counter BW)
K	64 to 1024 pixels per sampling block
n	~1Kbits, to be computed from Equation (B.9)
P	2 to 16 processors

B.1.3.3. Processor Timing Analysis. In any realistic analysis of computational cost, processor timing, and system performance, one must estimate the computational delay incurred by the processor itself. In the preceding model, this delay is represented by the variable $\Delta t_{P/DS}$. Assume that the counter-based model of D_H computation forms the basis for our original formulation of the architectural design shown in Figure B.1. Further assume that there are K pipelines per processor P_i, each of which implement the previously-discussed counter process. We initially make this assumption to show that computation of TNE-based pattern matching could be with K-fold parallelism. The following three observation-example pairs show that this assumption is untenable in practice for the worst case, where the sampling block and AM are fully processed. However, as shown in Sections B.1.3.3.7-8, it is possible, with partial or full pipelining of AM data through the counter, to process n/*p* columns of the AM at K pixel shifts of each input sampling block. This holds especially well for $f_K, f_L < 1$.

B.1.3.3.1. Observation. We begin this analysis of processor complexity by noting that, if the topologically longest path through the m-bit counter consists of N_{pg} gates, then the delay in non-pipelined processing of each AM column would be computed as

$$\Delta t_{P/DS} = K \cdot N_{pg} \cdot \Delta t_{clock}, \qquad \text{(B.11)}$$

where the variable Δt_{clock} denotes one clock cycle delay in P_i.

B.1.3.3.2. Example. If an FPGA is employed with 100MHz clock, then $\Delta t_{clock} = 10$ ns, which implies that, for a counter with longest path of 32 gates ($N_{pg} = 32$) and a K = 256 element sampling block, then $\Delta t_{P/DS} = 256(32)10 \text{ ns} \approx 8.2 \times 10^{-5}$ sec, or 82 μs. This implies that the temporal delay component of Equation (B.10), given by $n \cdot \Delta t_{P_iDS}$, equals approximately 84ms for a 1024-column AM partition computed on P_i.

For a frame rate of F = 1fps, this would mean that, given worst-case constraints $f_K = 1$ and $f_L = 1$ with image resolution of $N = 1024^2$ pixels, then $f_N = 1.14 \times 10^{-5}$, or 0.00114 percent of source pixels are interrogated. This statement can be verified by observing that

$$f_N = \frac{1}{1024^2 \text{ pixels/fra me} \cdot 84 \text{ msec/pixel} \cdot 1 \text{ frame/sec}},$$

Note that, if $f_K = 0.5$ and $f_L = 0.1$, as discussed previously, f_N would still equal 2.28×10^{-4}, which means that only 240 source pixels would be interrogated by TNE/ATR in the preceding case, which we implicitly assume (for purposes of this analysis) to be the best case. However, this assessment may vary in practice.

B.1.3.3.3. Observation. The preceding values for f_N are both infeasibly low, and derive from a value of Δt_{P_iDS} that has been set artificially high due to the unrealistic assumption of non-pipelined execution. Since we are assuming that the FPGA or RC processor P_i is a clocked (synchronous) circuit, it is not unreasonable to assume that, in the absence of set-up time, an N_{pg}-gate processor path could be cleared in N_{pg} clock cycles, filled in N_{pg} cycles with partially processed results, and require K cycles for execution of the counter process, thereby yielding a total time delay that can be expressed in terms of partially pipelined execution in terms of an execution time

$$\Delta t^{pp}_{P_iDS} = (K + 2N_{pg}) \cdot \Delta t_{clock} \,. \qquad \text{(B.12)}$$

This represents a speedup of

$$?_{pp} = \frac{?t_{P_iDS}}{?t^{pp}_{P_iDS}} = \frac{K \cdot N_{pg} \cdot ?t_{clock}}{(K + 2N_{pg}) \cdot ?t_{clock}} = \frac{K \cdot N_{pg}}{K + 2N_{pg}} = \frac{K}{K/N_{pg} + 2} \approx N_{pg}\,, \qquad \text{(B.13)}$$

where the approximation is valid for $N_{pg} >> 2$.

B.1.3.3.4. Example. If K = 256, $N_{pg} = 32$, and $\Delta t_{clock} = 10$ ns, then

$$\Delta t^{pp}_{P_iDS} = (K + 2N_{pg}) \cdot \Delta t_{clock} = (64 + 256)10 \text{ ns } = 3.2\ \mu s\,,$$

which implies that $n \cdot \Delta t_{P_iDS} \approx 3.3$ ms for a 1024-column AM partition computed on P_i.

Following Example B.1.3.3.2, given $f_K, f_L = 1$ and $N = 1024^2$ pixels, this implies $f_N \approx 2.89 \times 10^{-4}$ or 0.029 percent of source pixels are interrogated. If $f_K = 0.5$ and $f_L = 0.1$, then $f_N \approx 5.78 \times 10^{-3}$, or 0.578 percent of source pixels are interrogated by the TNE ATR algorithm. In a 1024^2-pixel image, this equates to 6,061 candidate target pixels.

B.1.3.3.5. Observation. The preceding situation can be further improved by not clearing the pipeline at the arrival of each AM column corresponding to an input pixel, except to remove possible spurious bits before the first sequence of K bits (the first column of each agreement map partition) is processed. This implies that full pipelined processing of a K-row by N-column AM partition would require time

$$T^{fp}_{exe} = n \cdot \Delta t^{fp}_{P_iDS} \equiv (Kn + 2N_{pg}) \cdot \Delta t_{clock} \text{ cycles}\,, \qquad \text{(B.14)}$$

which represents a savings of $(n-2) \cdot N_{pg}$ cycles, but a speedup that remains as

$$?_{fp} = \frac{n \cdot ?t_{P_iDS}}{T^{fp}_{exe}} = \frac{Kn \cdot N_{pg} \cdot ?t_{clock}}{(Kn + 2N_{pg}) \cdot ?t_{clock}} = \frac{Kn \cdot N_{pg}}{Kn + 2N_{pg}} = \frac{Kn}{Kn/N_{pg} + 2} \approx N_{pg} \qquad \text{(B.15)}$$

as in Observation B.1.3.3.3. However, the reliability of this circuit could be compromised by lack of clearing the counter prior to processing of each column corresponding to an input pixel.

B.1.3.3.6. Example. Selecting again the values K = 256, $N_{pg} = 32$, and $\Delta t_{clock} = 10$ ns, we obtain the following result for $f_K, f_L = 1$:

$$T_{exe}^{fp} = (Kn + 2N_{pg}) \cdot \Delta t_{clock} = (256(1024) + 64)10 \text{ ns } = 2.63 \text{ ms},$$

which represents a 25.5 percent speed increase over the value of $n \cdot \Delta t_{PiDS}^{pp} \approx 3.3$ ms computed in Example B.1.3.3.4.

Similar to Exampled B.1.3.3.2 and B.1.3.3.4, this implies that $f_N \approx 3.62 \times 10^{-4}$ or 0.036 percent of source pixels are interrogated. If $f_K = 0.5$ and $f_L = 0.1$, then $f_N \approx 7.2 \times 10^{-3}$, or 0.72 percent of source pixels are interrogated. In a 1024^2-pixel image, this means that 7,549 pixels would be interrogated as candidate targets. This equates to approximately one target of size 128x64 pixels, seven targets of size 32x32 pixels, or 29 16x16-pixel targets, which are realistic expectations for small-target imagery.

B.1.3.3.7. Observation. The numbers of targets discussed at the conclusion of the preceding example are sufficient, but are barely adequate for fields that contain a large number of targets (e.g., terrestrial surfaces containing broadcasted anti-personnel mines). The reason for this small number of targets (and large value of the product $n \cdot \Delta t_{PiDS}$) is that we have assumed (at the outset of this subsection) that there are but K counter pipelines available per processor (e.g., the case of a fast serial DSP). If an m-bit counter requires fewer than 6m gates (as in the case discussed in Section B.1.2.4), and incurs a longest-path delay of $N_{pg} < 3m$ gates, then this implies that, for m = 8, fewer than 50 gates per counter would be utilized, for a total of fewer than 12,800 gates if K = 256. By inspection of Equation (B.4), it is possible to implement many more than K such counters in a 100,000-gate FPGA. Hence, it is reasonable to consider the presence of *p* >> K pipelines on-chip.

In order to address the case of parallel (and, possibly, reconfigurable) processors P_i we modify that assumption to assume the implementation of $N_{cp} >> K$ pipelines per processor, which describes the case of fine-grained parallelism (as opposed to the term P used to designate degree of coarse-granular parallelism expressed as the number of processors). In this case, the time estimates in Equations (B.11, B.12, B.14) can be scaled quasilinearly with parallelism per pixel shift $p = N_{cp}/K$ pipelines available per pixel shift, to provide a cost estimate for processing K pixel shifts of a given sampling block as

$$n \cdot \Delta t_{PiDS} = n \cdot \left\lceil \frac{K}{p} \right\rceil \cdot N_{pg} \cdot \Delta t_{clock}, \qquad \text{(B.11')}$$

$$n \cdot \Delta t_{PiDS}^{pp} = \left\lceil \frac{n}{p} \right\rceil \cdot (K + 2N_{pg}) \cdot \Delta t_{clock} \qquad \text{(B.12')}$$

$$T_{exe}^{fp} = (K \cdot \left\lceil \frac{n}{p} \right\rceil + 2N_{pg}) \cdot \Delta t_{clock}, \qquad \text{(B.14')}$$

The following example calculates execution time for the preceding expressions.

B.1.3.3.8. Example. Let $N_{cp} = 2048$ pipelines per processor, and assume (for purposes of simplicity) that the requisite I/O bandwidth is available to circumvent on-chip I/O bottlenecks. As before, let K = 256, n = 1024, $N_{pg} = 32$, and $\Delta t_{clock} = 10$ ns, which implies that $p = N_{cp} / K = 2048/356 = 8$. The following results pertain:

$$n \cdot \Delta t_{P_iDS} = 1024 \cdot \frac{256}{8} \cdot 32 \cdot 10 \text{ ns} = 10.5 \text{ ms}$$

$$n \cdot \Delta t^{pp}_{P_iDS} = \frac{1024}{8} \cdot (256 + 64) \cdot 10 \text{ ns} = 0.409 \text{ ms}$$

$$T^{\mathfrak{p}}_{exe} = (256 \cdot \frac{1024}{8} + 64) \cdot 10 \text{ ns} = 0.328 \text{ ms}.$$

In the latter case, the speedup equals 2.63ms/0.328ms = 8, which is exactly p. Hence, it is possible to remedy the problems of low speed without resorting to source block subsampling (i.e., setting $\mathfrak{f}_K < 1$). The resulting speedup due to hierarchical partitioning of the AM column coordinate, which we have shown could approximate $\mathfrak{f}_L = 0.1$, would further increase the efficiency of processing.

B.1.3.3.9. Remark. In order to further increase system throughput in terms of number of source pixels interrogated, we foresee the employment of three potential technologies, which are:

1. *Faster FPGA or RC clock speed* – For example, Xilinx is developing a 5 million gate FPGA with estimated clock speed of 1 GHz. If current technology trends in FPGA development mirror the speed increases realized thus far in SISD CPU development, then one can expect FPGA clock rates in excess of 3 GHz within the next three years. In that case, the values of $\mathfrak{f}_N$ (and, concomitantly, the values of $n \cdot \Delta t_{P_iDS}$) would be increased (decreased) by a factor of over 1.5 orders of magnitude in comparison with values derived in Example B.1.3.3.6. This would potentially improve target acquisition capabilities under the preceding assumptions to over 2,000 16x16-pixel targets, or more than 170 32x32-pixel targets. Alternatively, $\mathfrak{f}_K$ could be increased for greater statistical coverage, and different partitioning schemes that might employ higher resolution in the template domain (e.g., $\mathfrak{f}_L$ more closely approaching unity) could be employed.
2. *Higher degrees of on-chip parallelism* – Given an increasing number of gates in FPGA designs (e.g., 20 million-gate FPGAs are expected in commercial quantity within five years), more pipelines could be replicated on-chip, thus facilitating increased parallelism in AM column input. This would decrease the degree of interleaving or replication of the delay $n \cdot \Delta t_{P_iDS}$ required for ATR under the prespecified constraints of 1fps and 1Kx1K-pixel imagery, making n-fold parallelism on-chip a reality, subject to the constraints of the von Neumann bottleneck between P_i and M_i.
3. *Increased I/O parallelism through optical backplane interconnects* – Although not a likely development in FPGA technology in the proposed Phase-II project period, it is reasonable to assume that current experiments that attempt to increase broadcast scope and speed in closely-coupled parallel processors could be eventually migrated to FPGAs, but on a very small scale. Chief among the technologies for large-scale instruction and data broadcast in electro-optical backplanes is multi-way communication facilitated by such innovations as wavelength-division multiplexing, spread-spectrum encoding, and highly miniaturized array detectors.

B.1.4. Algorithmic and Implementational Issues. In addition to the cursory analysis presented in the preceding sections, we discuss two technical issues of interest to the sponsor (FTI). Initially, we present a brief comparison of computational cost for different pattern matching approaches that appear to be significant to the proposed Phase-II effort.

The Hamming-like method of pattern matching discussed in Section B.1.2 is neither the only nor the optimal method for matching sampling block contents to a template database. For example, the Euclidean distance between two Boolean feature vectors can be computed from D_H.

It is interesting to note that, in the case of comparing Boolean vectors, the Hamming and Euclidean distances can both be computed via the XOR operation.

For example, given two Boolean vectors $\mathbf{a} = (a_1, a_2, \ldots, a_K)$ and $\mathbf{b} = (b_1, b_2, \ldots, b_K)$, the Hamming distance can be expressed as:

$$D_H = \sum (\mathbf{a} \text{ XOR } \mathbf{b}),$$

while the Euclidean distance

$$D_E = \sqrt{\sum_{i=1}^{K} (a_i - b_i)^2} = \sqrt{\sum (\mathbf{a} \text{ XOR } \mathbf{b})} = \sqrt{D_H}$$

can be computed from the Hamming distance external to P_i (e.g., in the FPGA host). A similar comment holds for the RMS distance

$$D_{RMS} = \sqrt{\frac{1}{K} \cdot \sum_{i=1}^{K} (a_i - b_i)^2} = \sqrt{\frac{1}{K} \cdot \sum (\mathbf{a} \text{ XOR } \mathbf{b})} = \sqrt{D_H / K} \ .$$

As a result, the summation of XOR results constituent to the Euclidean and Hamming distances would require the same computational cost in the architecture of Figure B.1, regardless of which of these two distances was being computed.

It is important to note that the preceding concept is relevant for Boolean vectors only. For real-valued vectors **a** and **b**, the Hamming distance is undefined unless a thresholding operation or other type of comparison operation is introduced, which would yield a Boolean result.

Additionally, we note that matching of real-valued vectors on-chip would require integer or floating-point addition and multiplication at the very least, both of which are space-consumptive and tend to result in unfeasibly slow circuits for current FPGA implementations. For example, given a pattern class **p** and its covariance matrix **C**, together with a mean-pattern **m** comprised of K elements, the Mahalanobis distance between a K-element test vector **a** and **m** would be computed as

$$D_M = (\mathbf{a} - \mathbf{m})' \mathbf{C} (\mathbf{a} - \mathbf{m}) \ ,$$

which implies multiplication of a 1xK-element vector with a KxK-element matrix to yield a 1xK-element vector that is then multiplied by a Kx1-element vector to yield a scalar. This requires $\mathbf{O}(K^2)$ multiplications, which should be floating point operations if **p** is large (to achieve sufficient discrimination between patterns represented in **C** and **m**.

In a Phase-II study, we propose to model FPGA-based architecture performance in terms of I/O bus bandwidth and collision effects associated with realistic packet size (e.g., several columns of a TNE agreement map per packet). Additionally, we propose to model the effect of packet transmission errors and retransmissions on architecture performance, as well as MNN-directed TNE classification accuracy.

In the former case (performance), bus collisions can significantly reduce external I/O bus bandwidth, creating an I/O bottleneck that effectively reduces CPU utilization, thereby penalizing processor performance. In the latter case (classification accuracy), if information from TNE's binary pointer table contained in the agreement map is corrupted when transmitted over external or internal databus(es), then the AM would no longer contain correct partial classification results. Whether or not MNNs have sufficient noise tolerance to maintain classification accuracy (and, therefore, Pd and Rfa) in an MNN-directed TNE paradigm with significant bus errors is unknown. Additional research is therefore indicated in performance simulation for TNE architectures, as well as the incorporation of fault models and fault isolation procedures in reconfigurable processors with different types of external buses (e.g., PCI-bus versus Myrinet).

B.2. ATR Performance Metrics for MNN-Directed TNE

In previous research, the authors have employed several measures of Pd and Rfa, which are (a) computed componentwise, (b) area-based, or (c) pixel-based. The componentwise Pd is computed as follows:

Step 1. Apply an ATR filter such as the ADGF to a test image, then threshold or otherwise filter the result to obtain pixels deemed significant, which comprise an image **b**. Morphological filtering can be used in this step, to reduce the number of noise components whose size is well outside the range of expected target size.

Step 2. Sequentially label the connected components in **b** to produce an image $\mathbf{c} \in \mathbf{Z}_m^{\mathbf{X}}$ (not to be confused with a compressed image **c** discussed in Section 2). Given a truthing image **f** $\in \{0,1\}^{\mathbf{X}}$, which denotes target position and extent by unitary values, compute the fraction of pixels that are correct detections as

$$F_d = \frac{\sum \mathbf{b} \cdot \mathbf{f}}{\sum \mathbf{f}}, \tag{B.16}$$

and the fraction of pixels on target in **f** that are false alarms in **b**, as follows:

$$F_{fa} = \frac{\sum \mathbf{b} \cdot \bar{\mathbf{f}}}{\sum \mathbf{f}}, \tag{B.17}$$

where $\bar{\mathbf{f}}$ denotes the complement of **f**.

Step 3. Assume that we have a function $f_C : \{0,1\}^{\mathbf{X}} \rightarrow \mathbf{N}$ that counts the number of connected components in a Boolean image on **X**. For example, in Step 2, $n = f_C(\mathbf{b})$. Compute the probability of detection P_d from the number of components $N_t = f_C(\mathbf{f})$ and the number of components N_d that have common pixels in **b** and **f**, as

$$P_d = max(1, N_d / N_t) .$$

The upper limit of Pd is clamped to the unitary value, since Pd is a probability and there can be more than one target component in **b** per target component in **f**. One similarly computes the number of false alarms as the number of components N_f that have one or more pixels located in **b** but have no pixels at the corresponding location(s) in **f**. We can express this in terms of image algebra, as follows:

$$N_f = f_c(\mathbf{b} \cdot \bar{\mathbf{f}}) .$$

Observe that the number of detected components (e.g., N_t, N_f) will depend on the detection technique, in particular, upon the postprocessing operations that are applied to the output of the ADGF. In practice, we apply morphological filters to the thresholded ADGF output, to reduce the incidence of spurious noise.

One of the problems associated with componentwise Pd and Rfa metrics is computational cost – given an N-pixel image, connected component labeling can require as many as $O(N^2)$ operations for serpentine components. As a result, we prefer to approximate P_d and N_f by F_d and F_{fa}, which can be computed on parallel processors, as shown in the vector-parallel formulation of Equation (B.16) and (B.17). In preliminary tests, we have found that F_d and F_{fa} approximate Pd and N_f (in the latter case, with some manipulation) to within 10 to 15 percent error in imagery of natural scenes with sparse convex targets.

In a Phase-II study, we propose to analyze ROC curves constructed from F_d and F_{fa}, to determine their correspondence with ROC curves constructed from P_d and Rfa (derived from N_f). We also plan to show that F_d and F_{fa} are much more feasible computationally for real-time detection of performance deficits in support of dynamically adaptive MNN-directed TNE.